REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 3 Nov 95	3. REPORT TYPE AND DATES COVERED
4. TITLE AND SUBTITLE A New Model For Scheduling Radio Networks		5. FUNDING NUMBERS
6. AUTHOR(S) Mark L. Huson		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AFIT Students Attending: Arizona State University		8. PERFORMING ORGANIZATION REPORT NUMBER 95-25D
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) DEPARTMENT OF THE AIR FORCE AFIT/CI 2950 P STREET, BLDG 125 WRIGHT-PATTERSON AFB OH 45433-7765		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release IAW AFR 190-1 Distribution Unlimited BRIAN D. Gauthier, MSgt, USAF Chief Administration		12b. DISTRIBUTION CODE
13. ABSTRACT *(Maximum 200 words)*		
14. SUBJECT TERMS		15. NUMBER OF PAGES 117
		16. PRICE CODE

19960104 152

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

A NEW MODEL FOR SCHEDULING RADIO NETWORKS

by

Mark L. Huson

has been approved

July 1995

APPROVED:

[illegible], Chairperson

[illegible]

[illegible]

[illegible]

[illegible]

Supervisory Committee

ACCEPTED:

[illegible]

Department Chairperson

[illegible]

Dean, Graduate College

A NEW MODEL FOR SCHEDULING RADIO NETWORKS

by

Mark L. Huson

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

ARIZONA STATE UNIVERSITY

August 1995

Abstract

This dissertation explores the problem of radio network scheduling using graph models. It is well known that network scheduling problems map into the set of graph coloring problems. These problems are known to be NP–complete for arbitrary graphs, but can be optimally solved for some restricted classes of graphs.

The graph models previously used for radio networks are shown to be inaccurate in their representation of the problem domain characteristics that affect network scheduling. Therefore, a new restricted class of graphs, called *Point Graphs*, is defined that accurately models radio networks. This model suggests new approaches to solving the scheduling problem.

The graph coloring problems are then analyzed with respect to the problem domain. Two types of network schedule are common, broadcast and link schedules. These scheduling problems are solved using distance–2 vertex coloring and a variant of edge coloring called prn–scheduling. Traditional vertex coloring is defined only for undirected graphs. However, for network scheduling the arc direction is important. Therefore, a new problem, directed distance–2 vertex coloring, is defined, and shown to produce schedules which are superior to those produced using undirected coloring.

Within the context of broadcast scheduling it is proven that, despite being a restricted class of graph, both the traditional coloring and the new directed coloring problems are NP–complete for point graphs. An exception is a subset of point graphs called *linear point graphs* that can be colored in polynomial time. These complexity results provide the impetus for developing new approximation algorithms for graph coloring using features unique to point graphs. The performance of these algorithms is compared to the performance of existing algorithms.

An analysis of prn–coloring for point graphs reveals that this problem is also NP–complete. New approximation algorithms for prn–coloring are developed and their performance is compared to the performance of existing algorithms.

The use of the broadcast scheduling algorithms for distributed applications is then investigated and a new distributed algorithm is proposed for scheduling in networks composed of mobile transceivers.

Acknowledgements

There are many people without whom this dissertation never would have seen the light of day. First I would like to thank my chairman, Dr. Arunabha Sen, for his patience and support. Thanks to the members of my committee for their commitment and dedication. Also to the people in the Computer Science and Engineering Department office, especially Darlene for all the help she gave me with questions about school and the the paperwork involved in three years of graduate work. My thanks also go to the Department of Computer Science at the U. S. Air Force Academy. Especially LtCol Dino Schweitzer who selected me to teach at the Academy. It was that initial selection which led me here. I thank my parents, who provided inspiration through their support and example of pursuing post graduate education. My siblings were also there supporting me in spirit. My thanks go to the OCI group at the Community of the Blessed Sacrament, they were there to help me keep things in perspective and to recharge my batteries. In particular, my sponsor, Jack English and his wife Eleanor and Fr. Scott Ritchey. The others are too numerous to mention, but they have my heartfelt gratitude and payers. I also have a special thank you for Eileen R. Matty. She helped me stay in one piece during my time in Arizona. Finally, I'd like to thank the people who provided me with diversions from work. Reneé Hovden, and Cherié Rodgers all extended our friendship beyond the environment where we met and helped make my stay here tolerable. Though she probably doesn't fully realize it, Julie Madler prompted me to learn nearly as much about myself as I learned about anything else while I was here. I will miss them all, and will send them my prayers, love, and gratitude for the rest of my life.

Table of Contents

List of Tables

List of Figures

Figure Page

Chapter 1

Introduction

As we approach the twenty–first century it is said that man is entering the "information age." At the dawning of this new age, huge amounts of information are being generated and transmitted between individuals, organizations, and governments. The advent of computers and electronic communications has accelerated both the generation and dissemination of information within computer *networks.*

These networks are composed of computers which communicate using a shared physical medium. Most current networks use a hard wire connection which requires constant physical contact between the computer and the medium. These networks use different protocols such as ethernet or broadband transmissions to send information between computers (and other users) on the network. The rapid growth in computer and communications technology has led to an explosion in the demand for network connections and network services. In addition, the growth of mobile communication systems (cellular phones in particular) has resulted in a mobile work force and the demand for mobile network connections.

Mobile network connections (which will hereafter be referred to as mobile networks) rely on the use of wireless communications. Communication is usually achieved using radio transmissions as the shared medium within the network. In order to facilitate the high volume of information to be communicated within these networks, the radio transmissions must efficiently utilize the medium. Efficient utilization is dependent upon access to the shared medium. This can best be accomplished by understanding the problem and arriving at a solution based on the problem. The problem of shared access results in the need for scheduling access to the shared communication medium to ensure communication needs are met. This dissertation addresses the problem of understanding the scheduling access by investigating the process of modeling the problem domain and the efficiency of arriving at solutions using the models developed.

The rest of this chapter introduces some of the concepts and concerns involved in radio communication. Section 1.1 discusses radio network applications and development. Section 1.2 addresses some of the specific concerns applicable to the problem of accessing a shared communication medium. Section 1.3 gives a brief description of the reasons for and the approach to the work presented in this dissertation. Section 1.4 gives a brief description of the problem and the objectives of this dissertation. Finally, section 1.5 outlines the remainder of this dissertation.

1.1 Radio Communications

Radio communications occur by encoding a message on a signal within the electromagnetic spectrum. What distinguishes one radio signal from another is the method used to encode the message and the nature of the signal. Common encoding techniques are the amplitude modulation (AM) and frequency modulation (FM) common to broadcast radio around the world. The other feature separating these two forms of communication in the United States is the nature of the signal. For these two cases, the frequency of the carrier signal is used to separate the "AM band" from the "FM band".

Message encoding depends in part on the type of signal being encoded. For example, while the acoustic waveform is used to modulate the carrier signal in AM radio, the encoding of television signals is more complex due in part to the additional information being encoded. In fact, the type of information being sent as a message is not entirely independent of the signal used.

The signal frequencies used for radio communications range from approximately 10 kilohertz to 1 terahertz (near infrared). The difference between frequencies affects the amount of information they can carry in a given amount of time. Most commercial communications, including broadcast radio, television, aircraft, police, and maritime voice communications lie in the 0.5–300 MHz range. However, there is a great deal of growth in the usage of the frequencies beyond 300 MHz for applications such as satellite communication and land base microwave relay technology. As a rule of thumb, the more information per unit time, the higher the frequency needed to carry the information.

Radio transmissions are also prone to several limitations which affect the design of radio networks. Of particular concern are noise, interference from other signals, geographic interference, and jamming. These problems affect the performance and connectivity of networks. Two additional factors which influence the connectivity of a radio network are that signal strength diminishes proportionally to the square of the distance between the transmitter and the receiver and that many of the frequencies used for communications require a "line of sight" between the transmitter and the receiver. For more information on the characteristics of radio communications the reader is referred to any one of the quality texts on electromagnetics, communications, or networks [7, 14, 42].

Other applications of radio communications include situations where users are mobile, are in inaccessible areas, or when networks must be constructed quickly [47]. The spatial distribution of transceivers results in what is called a multi-hop network (i.e., communication between transceivers may require transceivers located between the source and destination of a message to receive and rebroadcast the communication). This distribution permits sharing or reuse of the frequencies. Current techniques for this sharing include *Time Division* [20, 47, 51], *Frequency Division* [30], and *Code Division* [34] multiple access schemes (referred to as TDMA, FDMA, and CDMA respectively).

1.2 The Need for Scheduling

The nature of radio broadcasts causes the problem of interference to be a concern within the radio network. The interference problem is what leads to the necessity of developing schedules for radio networks. Early in the development of radio it was discovered that transmitters assigned to the same or closely related frequencies, had the potential to interfere with each other. The first solution used was to combine the bandwidth of the transmitters and the band of the electromagnetic spectrum used to assign non-interfering frequencies to the transmitters. This is an early form of FDMA, and as long as the usable portion of the electromagnetic spectrum grew, unique frequencies could be assigned. Unfortunately, the proliferation of transmitters made this simple method of frequency assignment untenable. As spectrum usage, and transceiver sophistication grew, it became both desirable and necessary to provide an additional mechanism to prevent interference.

Regardless of the access mechanism used (TDMA, FDMA, or CDMA) the same basic problems persist. A *schedule* must be developed to allow all transceivers to function, and to prevent interference. The resulting schedule assigns a *position*, based on time, frequency, or code, to each transceiver. Each schedule position corresponds to an assigned time, frequency, or code in TDMA, FDMA, or CDMA respectively.

1.2.1 Interference Patterns

The type of schedule planned for a network affects the definition of interference. Two types of schedule are commonly used. A *broadcast* schedule determines a single schedule position for each transceiver. When a transceiver transmits a message, all transceivers within its range receive the message. In *link* schedules, transceivers are scheduled by message transmitter–receiver pairs, which is why it is often called point–to–point communication.

The type of schedule used depends on the inter–transceiver communications allowed in the network and, in the case of CDMA, on the code assignment scheme used. The capabilities of the transceivers determine (or are determined by) the access mechanism used. Typically, transceivers are assumed to be either transmitting or receiving at all points in the schedule (i.e., no transceiver does both simultaneously).

Two types of interference are of interest in developing network schedules. The first type is called *primary* interference. In broadcast scheduling, this occurs when two transmitters are within range of each other.

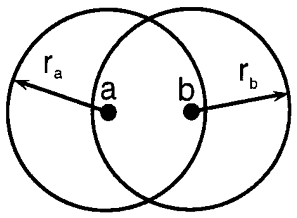

Figure 1.1: Primary Interference in Broadcast Scheduling

As shown in figure 1.1, transceivers a and b, with transmission radii r_a and r_b, are within each others' transmission range. We would not want them to attempt to transmit during the same time, using the same frequency or code, and would therefore schedule them to transmit in different positions in our schedule.

The second type of interference is *secondary* interference. Two transceivers should not transmit at the same position in the schedule if there is a third transceiver which is within the range of both transmitting transceivers.

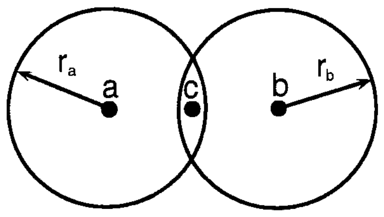

Figure 1.2: Secondary Interference in Broadcast Scheduling

Figure 1.2 shows that since transceiver c can receive transmissions from both a and b, a and b should not occupy the same schedule position.

When developing a link schedule interference occurs under slightly different conditions. *Primary* interference occurs when a transceiver is involved in the transmission (either as transmitter or receiver) for more than one message. Since transmissions are scheduled on a pairwise basis, this can correspond to a violation of either of the interference conditions from broadcast scheduling. Figure 1.3 shows the ways in which this can occur.

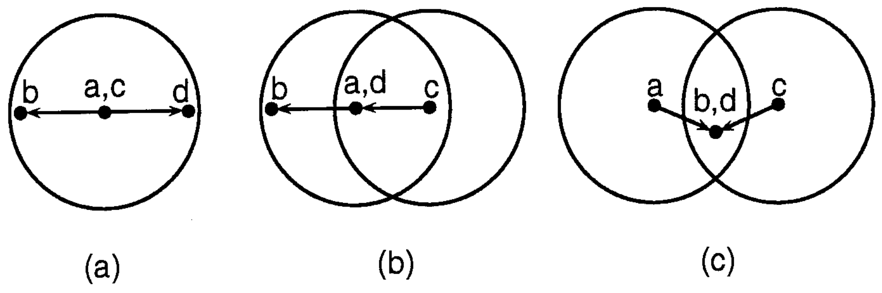

Figure 1.3: Primary Interference in Link Scheduling

For figure 1.3(a) the interference occurs because transceiver *a,c* cannot transmit two messages simultaneously. Likewise, for figure 1.3(b) transceiver *a,d* cannot both transmit and receive at the same time. This interference is equivalent to primary interference in broadcast scheduling. Finally, in figure 1.3(c), transceiver *b,d* cannot receive two messages at once, a condition equivalent to secondary broadcast interference.

Secondary interference in link scheduling differs from interference in broadcast scheduling in that there are unintended receivers for a given message (in broadcast scheduling it can be argued that all transceivers within range are intended to receive a transmission). This is demonstrated in figure 1.4. Due to the nature of electromagnetic transmission, most transceivers are omnidirectional, and though a transceiver may not be intended as the recipient, if it is within range and is not also transmitting, it will receive the message.

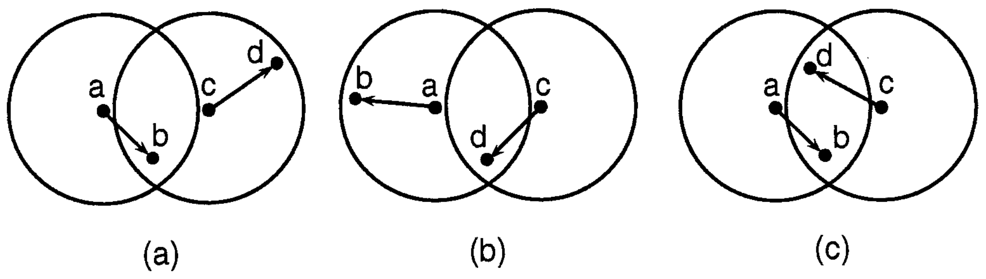

Figure 1.4: Secondary Interference in Link Scheduling

CDMA, due to differences in its transceiver model, treats this interference differently depending on the code assignment scheme used. Scheduling for a pairwise code assignment scheme is equivalent to the more traditional edge coloring problem, and therefore does not consider the secondary interference shown in figure 1.4 as a scheduling restriction [34]. This type of interference can be important for TDMA and FDMA networks and becomes an issue when hybrid code assignment schemes are used in CDMA.

One important note about the restrictions here, there is no restriction on simultaneous transmissions from two adjacent transceivers, as long as there are no common recipients for the transmissions (recall that the typical transceiver model prevents a transceiver from simultaneously transmitting and receiving). Thus, figure 1.4 could just as easily be drawn as in figure 1.5. While interference occurs in this figure, it is not due to the fact that transceivers a and c are within each other's range. If transceivers b and d were not within the common transmission area (the area where the circles overlap) then no interference would occur.

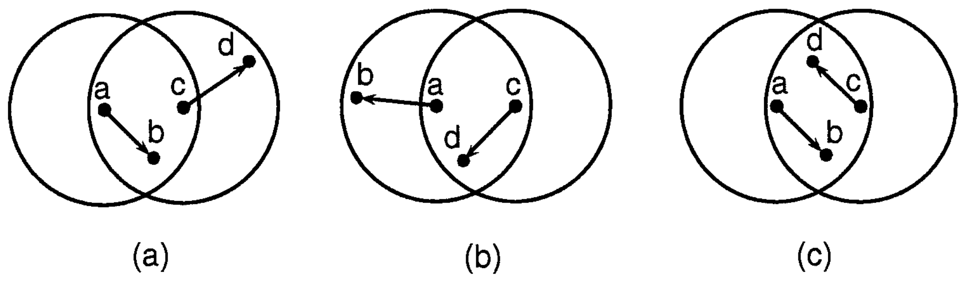

Figure 1.5: Secondary Interference in Link Scheduling, Redrawn

1.2.2 Effect of Interference on Scheduling

Interference between transceivers is the driving force behind network scheduling. Two approaches are commonly used in scheduling networks. *Random* access scheduling produces no set schedule per se, instead a transceiver transmits a message as soon as possible. If a collision occurs, the transmission is repeated until the message is successfully received. This method would be attractive in networks with infrequent communication because interference would be unlikely. *Random* access is the most common method for hard wire networks. For those networks, collisions can easily be detected since all the transceivers can communicate with each other. In addition, if a transceiver does not need to send a message it doesn't prevent other transceivers from transmitting in its schedule position. These advantages permit efficient use of the network medium by reducing the overhead involved in scheduling. As long as collisions between messages are rare, the overhead of collision detection will be less than the overhead of maintaining schedule positions which are infrequently used. *Random* access scheduling is common in networks which have bursts of communication rather than a constant amount of message traffic.

The alternative to *random* access "scheduling" is *fixed* access. The advantages of *random* access scheduling tend to be lost when using broadcast media. Much of the efficiency is lost due to difficulties in detecting collisions (the *hidden terminal problem* [48, 56]). On the other hand, a *fixed* schedule assigns a portion of the schedule to each transceiver. In broadcast networks, the spatial separation of transceivers permits the same position in the schedule to be assigned to multiple transceivers while avoiding collisions. Therefore, *fixed* schedules are common in broadcast networks. (In addition, in heavily loaded networks, communication is often improved when collision detection is not required.)

1.3 Impetus and Approach

Radio network scheduling has been shown to be one of the intrinsically hard problems in terms of the complexity of arriving at a schedule of minimum length. A nearly equivalent graph theory problem has been the main area of focus for research in generating approximate solutions in a reasonable amount of time. The research has been somewhat misdirected in that both the graphical models used and the specific graph theory problem do not accurately represent the radio networks being scheduled. This work proposes a more accurate graphical model, and defines a graph theory problem which captures the pertinent details lost in the traditional graph theory problem.

To present these arguments, the failings of the more popular models and the graph theory problem will first be presented. After defining the new graphical model and the graph theory problems used to solve scheduling, the complexity of solving both the traditional and new graph theory problem are examined, and new algorithms are developed which attempt to exploit the characteristics of the radio networks which were not captured previously.

1.4 The Problem and Objectives

Previous research addressed the wrong problem by using network models which don't reflect the possible interconnections between the transceivers in the network. The results using their models are related to the problem of scheduling radio networks, but due to their inaccuracy, the conclusions drawn are not directly applicable to the scheduling problem. A new model is needed which can accurately represent the problem domain of scheduling radio networks.

The main objective of this work is to define and investigate an accurate model of radio networks. Specific objectives can be defined as follows:

- Develop an accurate model of radio networks which can be used to solve the broadcast scheduling problem.
- Prove the complexity of solving the broadcast scheduling problem using the new model.
- Develop new algorithms to explore the characteristics unique to the new model, for possible exploitation in developing broadcast schedules.
- Show the correctness and complexity of the new algorithms.
- Empirically compare the performance of the new algorithms to the performance of algorithms which do not use the unique characteristics of the new model.
- Develop a distributed broadcast scheduling algorithm for possible use in networks composed of mobile transceivers.

1.5 Overview of Dissertation

Chapter 2 describes previous work in the area of scheduling radio networks. After a discussion of the ideas behind modeling network problems, the correspondence between network scheduling and graph coloring are explored. Typical graphical network models are evaluated with regard to their fidelity in representing the problem domain. In addition, the graph theoretic problems used to solve network scheduling using these models are discussed. Deficiencies in the models and traditional methods of solution are identified.

Chapter 3 defines a new graph model for radio networks to address the deficiencies identified in the previous chapter. The relationship of this model is shown with respect to the previous graphs used for this problem. The importance of redefining the methods of solution with respect to the problem domain is documented in terms of the efficiency of the solutions produced. For this reason, a new definition of the graph coloring problem is presented in this chapter.

Chapter 4 explores the process of solving the radio network broadcast scheduling problem. The traditional and new graph vertex coloring problems are investigated with respect to the complexity of solution. Due to the NP–completeness of optimal solutions, new approximation algorithms are developed which incorporate the additional information included in this model. The performance of these algorithms is analyzed with respect to their running time and performance relative to optimal solutions to the problem. Empirical performance of these new algorithms is compared to versions of traditional algorithms which are modified to address the newly defined problem.

Chapter 5 discusses the complexity of link scheduling by solving the prn–coloring problem for the new graph models. Approximation algorithms are defined which use the additional information available in the new graph model. The performance of these algorithms is compared to that of existing algorithms for determining link schedules.

Chapter 6 addresses distributed scheduling in radio networks. Transceiver capabilities within the network are analyzed in terms of their importance in distributed scheduling. An algorithm for scheduling in a distributed environment is presented along with a set of assumptions about the transceiver capabilities.

Chapter 7 summarizes the work presented in this dissertation. The major contributions of this dissertation are presented. New directions for research indicated by the results of this work are suggested.

Chapter 2

Background

This chapter begins with a discussion of the reasons for modeling networks in developing solutions to the problem of scheduling shared access to the common transmission medium. The correspondence to graph theoretic problems is then explained and the types of graphs used in developing solutions to the problem are examined. Next, the complexity of solving the graph problems is explored. The chapter concludes with a discussion of the limitations of the graphical models used and the graph theory problems solved in developing network schedules.

2.1 Reasons for Network Modeling

Models are designed to encapsulate the essential features of the system under study. Computer models must be constructed in terms of computable functions and, as such, require the adoption of a particular view or paradigm of the system. The resulting model represents the view of the "real world" or at least those aspects of interest in the problem under investigation. The main goal in developing a model is to derive a simple, easily manipulated representation which can be used to solve the problem. The keys are to capture the important aspects of the problem and to end up with a form of the problem which lends itself to solution.

The most important criteria in any network schedule is to have an interference free schedule. This means the network model used must be able to represent the transceivers to be scheduled and the various types of interference which may be present between them. With any transceiver there are certain features which are only of secondary importance in scheduling. (For example, while antenna height will affect the range of the transceiver, the range is determined by a combination of characteristics, of which antenna height is only one. Therefore, while range becomes a major consideration in determining if interference will occur between transceivers, antenna height is only of secondary interest.) For the sake of simplification, such secondary features are subsumed under larger aspects of the problem. Of course, the particular problem being solved will determine what features are of secondary importance, which ones are essential to the problem model, and which ones have absolutely no bearing on the model.

The goal is to have an accurate model of the system which can be used to develop solutions to the problem. The quality of the model is usually measured by its fidelity in capturing the information of the problem domain, while excluding representations not possible in the problem domain. The other main concerns are the efficiency involved in using the model with respect to the time required to arrive at a solution, and the quality of the solution (the quality of solution is influenced by both the model being used and by the algorithms used).

Most network models use a standard transceiver as the abstraction of all the transceivers in the network. Each transceiver differs from the others only in its position. While other transceiver models are possible, this simplifying assumption is nearly universal in its application to network modeling. This assumption imposes restrictions on the possible communication links between transceivers in the network, and will be relaxed as appropriate in discussing network models and algorithms for those models.

2.2 Correspondence to Graph Theoretic Problems

Graphical models are particularly attractive for network scheduling problems due to the similarity between scheduling and some graph theoretic problems. Network scheduling is usually done by solving graph coloring problems for the graphical representation of the network to be scheduled. Traditionally, graph coloring is concerned with assigning colors to the vertices of graphs so no two adjacent vertices have the same color. Adjacency in these cases has been restricted to a definition based on undirected graphs. By this definition, two vertices are adjacent if and only if there is an edge between them. For vertex coloring (distance–1 coloring) this definition is adequate to represent the restrictions placed on a graph coloring. Based on this definition, two vertices cannot be assigned the same color if they are adjacent.

For example, frequency assignment is a scheduling problem where interfering transmitters are assigned different frequencies. This problem is usually solved using distance–1 vertex coloring [30]. In it, two nodes u and v, are adjacent if $\exists e \in E$, the set of edges, where e is an edge between u and v. Such vertices are not allowed to have the same color in a correct coloring. Stated formally:

> Given graph G = (V,E), a *distance–1 vertex coloring*, also called a *graph coloring*, is a mapping $\{\Phi : v \rightarrow \mathcal{N} | \phi(v_i) = \phi(v_j) \Leftrightarrow (v_i, v_j) \notin E\}$ where $v_i, v_j \in V$.

While this graph coloring problem corresponds to the frequency assignment problem, most scheduling problems correspond to different graph theoretic problems. Broadcast scheduling in a multi–hop radio network can not be solved using distance–1 vertex coloring. The reason for this is that the definition of interference is different for these types of networks. For these networks, nodes u and v cannot have the same color if either:

1. u and v are adjacent or
2. $\exists w \in V$ where w is adjacent to both u and v

These conditions are shown in figure 2.1. This problem is called distance–2 vertex coloring and has been studied by several authors within the context of networks [20, 51, 47]. All of these authors used a network model which incorporated uniform range transceivers. The result being that adjacency in these models is a symmetric relation (i.e., if aRb then bRa). This issue will be discussed in section 2.5.

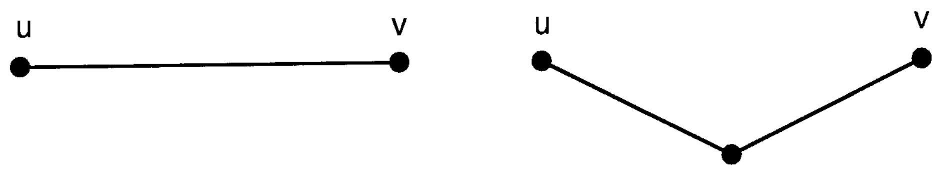

Figure 2.1: Conditions under which nodes u and v cannot have the same color in a distance–2 coloring of graph G.

Link scheduling is commonly used when the network uses routing information to transmit messages between transceivers which can only send messages through intermediate nodes. For link scheduling the issue of the access method becomes important in defining interference within the network [48, 33, 34]. Hu examines spread spectrum communication using Code Division Multiple Access with pairwise code assignment. This allows multiple messages to be received at a transceiver using orthogonal codes to discriminate between the received signals. When a strict pairwise assignment scheme is used, secondary interference (where a transceiver receives an unintended message) is not an issue, however, this is not true of "hybrid" code assignments [33, 34]. Pairwise code assignment reduces the constraints caused by interference within the network and allows traditional edge coloring to be used in assigning codes. Though edge coloring is an NP–Complete problem [32], efficient polynomial time approximation algorithms exist to determine near optimal edge coloring for arbitrary graphs. The worst case bounds for such algorithms is $O(1)$ in that the solution is guaranteed to be no more than the minimum (optimal) number of colors plus one. For this reason the primary focus of research has been on a more general problem where secondary interference, as described in chapter 1, is present. This is the link scheduling problem examined in [48, 47], called *prn–coloring* in those works. That terminology is adopted in this work to remain consistent with existing literature.

A formal definition of is:

> Given a graph G = (V,A), a *prn–coloring* of the graph is a mapping $\{\Phi : (a,b) \in A \to \mathcal{N} | \phi(a,b) = \phi(c,d) \Leftrightarrow a \neq b \neq c \neq d \text{ and } (a,d),(c,b) \notin A\}$.

Note that while there is a restriction on links between the transmitting and receiving transceivers in prn–coloring, there is no restriction on links from or between the receiving transceivers or the transmitting transceivers. Figure 2.2 depicts these two situations. In figure 2.2(a) the transmitting transceivers do not interfere, while in figure 2.2(b) the receiving transmitters do not interfere.

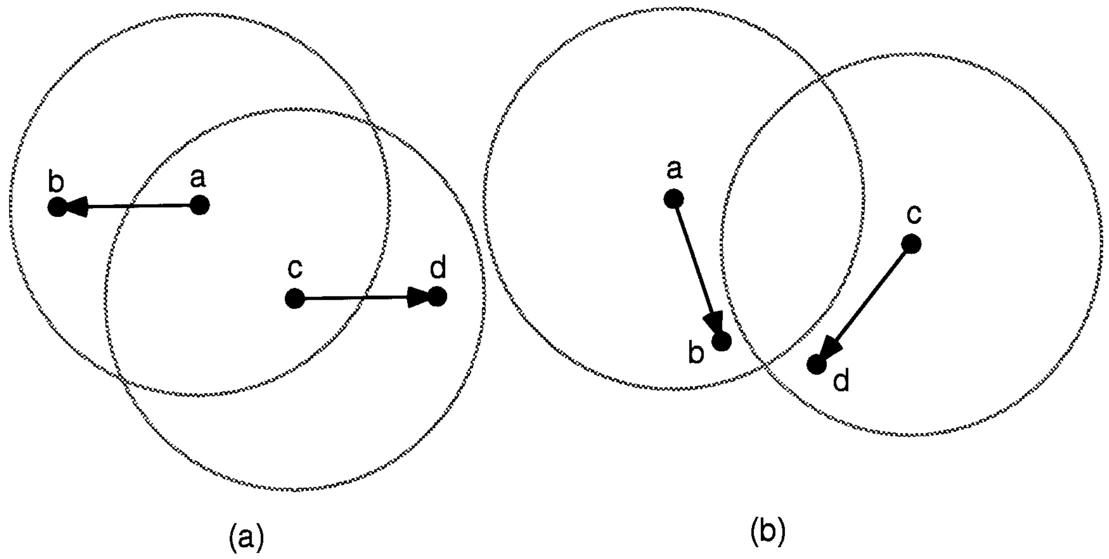

Figure 2.2: Non–interfering transmissions in link scheduling.

2.3 Graphical Models

Several graphical models have been used to represent networks. These graph models were all constructed by using the same construction method. The transceiver (or transmitter) positions and radii were used to define the graph vertices and edges respectively. The resulting graphs were then treated as one of the graph types used in modeling. Early network problems involved frequency assignment for radio and television stations, and these were modeled primarily by arbitrary graphs (though they were also modeled by *unit disk graphs* which will be defined later) [30]. As the focus shifted away from transmitters to transceivers, and the problem changed from a distance–1 to a distance–2 coloring problem for broadcast scheduling, restricted classes of graphs became attractive as network models. This was due to the existence of reasonable algorithms for solving graph coloring problems for these restricted graphs.

Three primary graph models have been proposed in the literature. The most restricted of these three, trees, is desirable because of the simplicity of coloring the graph. The other two predominant models are planar graphs and arbitrary graphs. It must be noted that most network investigators have concerned themselves with the algorithms used rather than the particular model used. Therefore, most existing work uses arbitrary graphs as the preferred network model.

A fourth graphical model has been used in the domain of the frequency assignment problem. Unit disk graphs define regions of interference for groups of fixed position transmitters in this problem domain. They have not been used in the more complex domain of network scheduling.

2.3.1 Trees

Trees are the simplest graphical representation used in modeling networks. There are many practical network problems which can be addressed using tree models (message routing and propagation, just to name a few). The use of trees as a model for solving network scheduling is fairly recent, and it is claimed that most existing packet radio networks can be modeled by trees [5].

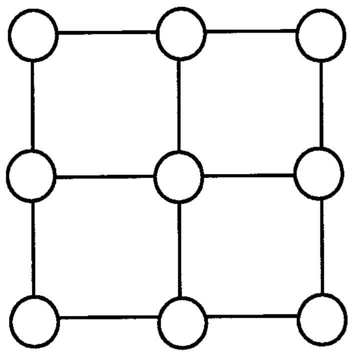

Figure 2.3: A network which cannot be accurately represented by a tree.

The main attraction of tree models is the simplicity of scheduling. A tree can be colored (and the corresponding network scheduled) using a minimum number of colors in polynomial time. This is true whether discussing distance–1 or distance–2 coloring. Unfortunately, trees are not flexible enough to represent many possible network configurations. Figure 2.3 is an example of a graph which has no accurate representation as a tree.

2.3.2 Planar Graphs

The limitations of trees led Ramanathan to propose planar graphs as the model of choice [48, 46, 47]. Planar graphs do provide more fidelity in representing the structure of radio networks. For example, the network in figure 2.3 is a planar graph. This added flexibility of representation does come at a cost because the algorithms for distance–2 coloring of planar graphs are more complex and require more time to execute than those for trees. The other problem with planar graphs is an inability to represent valid (and likely) network configurations. Just as with trees, this limits the applicability of planar graphs in solving radio network scheduling. The network in figure 2.4 is a simple example of a network which cannot be accurately represented using a planar graph.

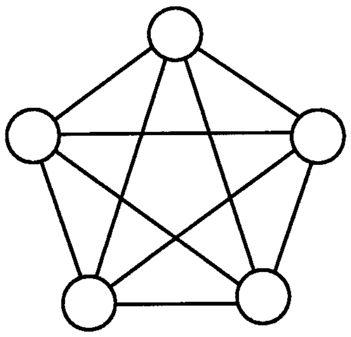

Figure 2.4: A simple network with no planar graph representation.

Ramanathan and Lloyd recognize this shortcoming of their model, and discuss *near–planar* graphs as being the practical model used in their work [48, 47]. In developing solutions using this model, the graphs are decomposed into *in* and *out* oriented graphs for link scheduling, and rely on bidirectional communication links in broadcast scheduling. Their approach changes the process of coloring a single graph,

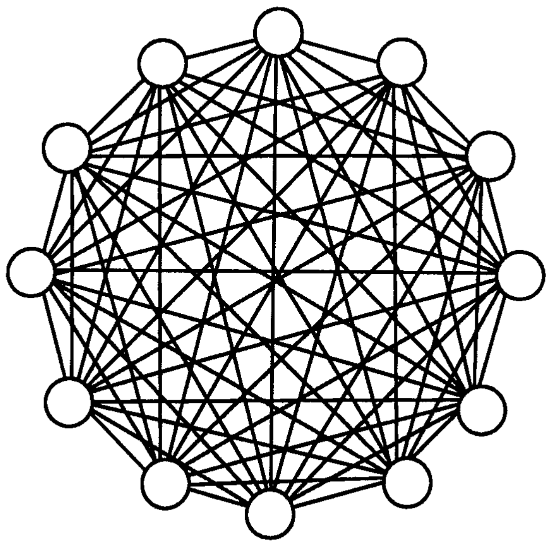

Figure 2.5: A network with poor planarity (high thickness), a possible result of increased network density.

to one of coloring a set of interrelated planar graphs. The measure of planarity they use is the thickness of a graph, which is defined as the minimum number of planar graphs into which a graph can be partitioned. It is precisely these problems which limit the application of planar graphs as models of radio networks.

Figure 2.5 shows that as the density of transceivers increases in any given area, the planarity of the graphical representation will decrease, further limiting the applicability of planar graph models to representing the network. With the projected increase in the usage of communication networks and no foreseeable change in the radio spectrum being used, planar models will be unable to accurately represent the networks being scheduled.

2.3.3 Arbitrary Graphs

If these restricted classes of graphs cannot accurately model radio networks, why use them? One alternative is to use arbitrary graphs to represent the network. The question is, how accurately do arbitrary graphs model radio networks. The reason for using restricted graphs is to use a model which gives a sufficiently accurate representation while illuminating some aspects of the problem structure which might help in solving the problem (and of course the hope of finding a model for which known polynomial time algorithms exist). Arbitrary graphs have been studied in [51, 20].

Arbitrary graphs have the advantage of being able to represent all possible network configurations. We have shown that this is not possible with the two other graphical models proposed in the literature. But how accurately do arbitrary graphs represent the network? One way of characterizing a class of graphs is to define a set of subgraphs called *forbidden subgraphs*. We used this idea in showing that trees and planar graphs do not accurately represent radio networks, by displaying network configurations which are forbidden subgraphs for each type of graph. In this section, we will show that networks can be modeled by a restricted class of graphs by showing some arbitrary graphs which have no real physical counterpart in radio networks.

Figure 2.6 shows several arbitrary subgraphs which have no physical representation in radio network topology. All the graphs in this figure have been identified as forbidden subgraphs of the class of graphs representing radio networks composed of uniform transceivers.

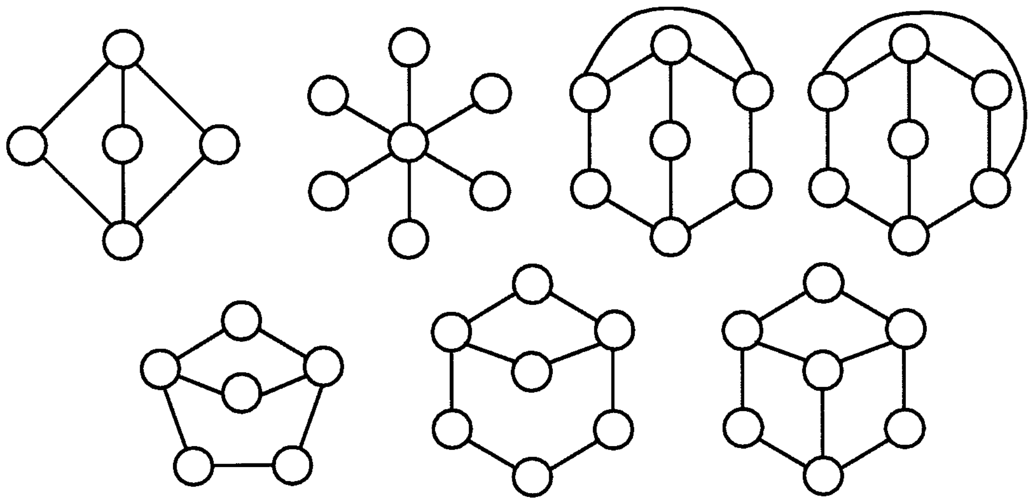

Figure 2.6: Forbidden subgraphs of the class of graphs representing radio networks consisting of uniform transceivers.

From this argument it is clear that arbitrary graphs do not capture the geometric limitations present in the physical system being modeled. Even if non–uniform transceivers are allowed in the model of the network, the resulting graphical representations will not be arbitrary. Figure 2.7 shows a simple example of a directed graph for which no corresponding physical network can be constructed.

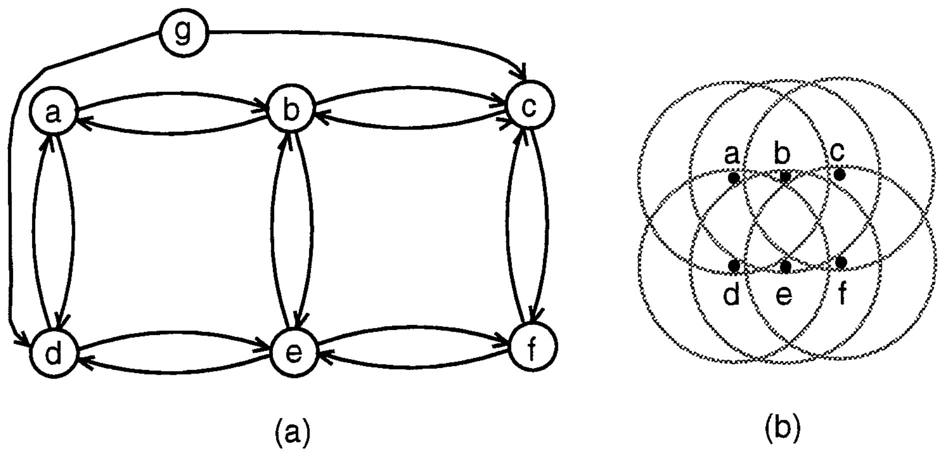

Figure 2.7: A simple directed graph with no corresponding physical network.

Note there is no combination of ranges and locations for transceivers a–f which leaves an overlap in the communication ranges of c and d which is not also contained in the communication range of one of the other transceivers. Therefore, there is no location for transceiver g to be placed in figure 2.7(b).

2.3.4 Unit Disk Graphs

Another type of graph which has been examined with regard to the frequency assignment problem is the unit disk graph [30]. This type of graph is of interest because it is a special case of the graphical model proposed in chapter 3. For the frequency assignment problem, this type of graph is constructed in the following manner. A disk of unit radius is placed on a plane at the position of each transmitter to be assigned a frequency (it is assumed all transmitters have the same range). A vertex in the unit disk graph appears for each transmitter, and transmitters t_i and t_j have an edge between them if their corresponding disks overlap. This is useful for the problem of frequency assignment since any receiver in the region of overlap would be within the transmission range of both t_i and t_j. Note that this model is based on the idea of radio transmitters rather than communicating transceivers in a network.

2.4 Complexity of Graph Coloring

The use of graphs to represent networks led to the use of graph coloring algorithms for scheduling and frequency assignment. Specifically, frequency assignment corresponds to distance–1 vertex coloring, broadcast scheduling in radio networks corresponds to distance–2 vertex coloring, while link scheduling corresponds to a variant of edge coloring called prn–coloring. Graph problems are known to be difficult problems, and NP–completeness proofs exist for the coloring problems of various restricted graphs. There are also specific types of graphs for which graph coloring (for example distance–1 vertex coloring of trees) is known to be solvable in polynomial time. One reason for pursuing this work is to determine if the type of graphs constructed for network modeling can be solved in polynomial time.

Optimality is an important issue in developing schedules. If just one extra schedule position more than the minimum (optimal) number is used every 5 seconds, 720 schedule positions are wasted each hour. Measured as a percentage, a schedule which is 1% longer than optimal wastes nearly 15 minutes a day. The NP–completeness of graph coloring for specific graph types means that approximation algorithms must be used to color the graph and determine a usable schedule. The amount of error in the approximate solution is a required trade off to obtain schedules for many networks in a reasonable amount of time [23].

Tree graphs are known to be colorable in polynomial time. This result holds for both distance 1 and 2 vertex coloring, and is the reason the Bar–Yehuda, et. al. [5] propose that all networks be modeled by trees. Similarly, edge coloring in trees can be done in polynomial time (polynomial on $|V|^2$ or $|E|$). Garey and Johnson show in [23] that graph K-colorability for arbitrary graphs is NP–complete for all K$\geq$3, and it remains NP–complete with K=3 for planar graphs having no vertex degree exceeding 4 [24]. Hale shows that vertex coloring for unit disk graphs is also NP–complete [30]. However, the problems we are looking at are distance-2 (2-hop) vertex coloring and edge coloring. Distance-2 coloring is not necessarily NP–complete based solely on the

NP–completeness of the vertex coloring problem.

There are various approaches to proving distance-2 vertex coloring NP–complete. For planar graphs, Ramanathan and Lloyd [46] proved the 7–planar–distance–2 coloring problem is NP–complete by construction from a k–distance–2 coloring proof for arbitrary graphs in [21], where $k \geq 3$.

Ramanathan uses component construction in his proof. The choice of planar graphs is due to the use of planar graphs as the network model in his work. In developing components for the polynomial time transformation of an arbitrary graph to a planar graph, the complexity of the edge crossover became the driving force in defining exactly what transformation was required. The crossover component in [48] required a minimum of 7 colors for a valid distance–2 coloring. This is why Ramanathan proved 7-planar–distance–2 coloring NP–complete.

Another, more general proof for arbitrary graphs appears in [51] and [20]. These proofs are based on an earlier proof by Ephremides and Truong which contained an error. The following is a sketch of this more general proof, using an undirected graph as the network model.

The method of proof is to use the network graph $G = (V, E)$, and construct a new augmented graph $G_a = (V_a, E_a)$ from G. New nodes are added to the graph, where each new node represents an edge from the set E. Then new node set is then defined by

$$V_a = V \bigcup \{v_{i,j} | (i,j) \in E\}$$

Edges are then added which connect the new node to the two vertices i and j, and to all the other newly added nodes. Therefore,

$$E_a = E \bigcup \{(v_{i,j}, i), (v_{i,j}, j)\} \bigcup \{(v_{i,j}, v_{k,l}) | (i,j), (k,l) \in E\}$$

Finally, another new graph, G', is then constructed from G_a such that $G' = (V_a, E')$ where $E' = E_a - E$. It is then shown that if there is a k'–distance–2 coloring solution for G', there is a k–coloring solution to G, where $k = k' - |V_a - V|$. For example, the graph, G, in figure 2.8(a) becomes the graph in figure 2.8(b).

Ramanathan and Lloyd also show the complexity of link scheduling using prn–coloring (the derivative of edge coloring discussed above) is NP–complete. Their

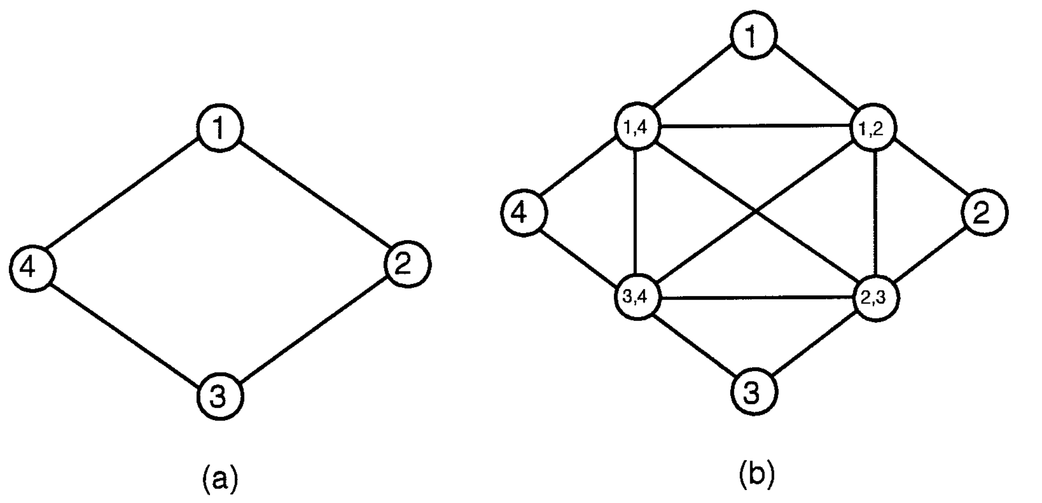

Figure 2.8: Parhi's transformation of G to G' for a given graph.

method of proof is by component replacement using the NP–completeness of 7–planar–distance–2 coloring to prove the complexity of 7–planar–prn–coloring is NP–complete [48, 46].

2.5 Model and Solution Limitations

The graph models used previously in examining and scheduling all lack fidelity in representing the problem. In addition, the graph coloring problems which have been used to solve the network scheduling problem using these graphs have been applied directly without regard to the problem domain being modeled.

2.5.1 Model Fidelity

As shown previously, tree, planar and arbitrary graphs all fail to capture the characteristics of the network. Both trees and planar graphs were shown to be too restrictive. They do not represent the problem domain characteristics which are important to scheduling. Common network configurations cannot be represented using these graphs. Arbitrary graphs, on the other hand, fail to capture the geometric limitations of the physical system within the model.

The transceiver interconnections are the characteristics of radio networks which are required to solve the scheduling problem. The graph models previously used either over constrain the problem by ignoring existing connections or consider too general a problem by not restricting interconnections at all. Essentially, the wrong problems were solved in previous work.

2.5.2 Undirected Graph Coloring

Graph coloring is not traditionally discussed for directed graphs, rather only for undirected graphs. The imposition of this restriction reduces the efficiency of graphical solutions to problems in some areas. In particular, multi–hop radio networks with transceivers of various ranges are over constrained.

For example, consider the situation in figure 2.9. In this case, transceivers B

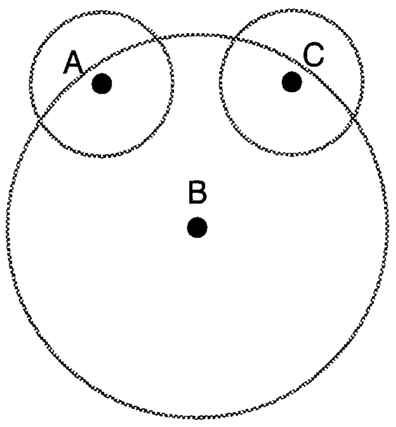

Figure 2.9: Transceiver locations with ranges highlighted.

can transmit to both A and C. However, neither A nor C can transmit to B, nor can they transmit to each other. Using the undirected graph model results in the representation in figure 2.10.

Based on this representation, a distance–2 coloring of this graph would require 3 colors, corresponding to 3 schedule positions. The traditional restriction of graph coloring problems to undirected graphs introduces additional constraints which increase

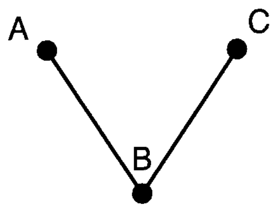

Figure 2.10: Undirected graph representation of figure 2.9.

the number of colors used. The resulting schedules are therefore clearly not optimal.

The usual practice is to construct a graph, $G = (V, E)$, from the network information. For each transceiver, t_i, in the network a vertex, $v_i \in V$, is defined. For every pair of transceivers, t_i, t_j, if the Euclidean distance $d(t_i, t_j)$ is less than or equal to the range of either t_i or t_j then an edge exists between v_i and v_j. Depending on what variations exist between transceivers within the network, a large number of non–existent communication links may be added as edges in creating the network model.

Chapter 3

Network Modeling and Graph Coloring

The discussion in chapter 2 showed the failings of previous network models and of the graph theoretic problems used to solve the network scheduling problem. This chapter presents a new model of radio networks which addresses the shortcomings of previous models. The graphical model, called a *Point Graph*, depicts transceivers by position and range. The graph theoretic problem of vertex coloring is then redefined in terms of directed coloring.

3.1 Point-Graphs: A New Model

It has long been recognized that networks can be described by a list of transceiver positions and ranges [30, 48, 50, 20]. These positions were used to define unit disk, planar, and arbitrary graphs respectively. No other information has ever been used in generating the graph models used to schedule radio networks. The reasons for the selection of a particular class of graph in modeling a network was driven by algorithms to perform graph coloring rather than the desire to accurately reflect the structure of the network being modeled. The model selected was then used to create solutions and to analyze the complexity of solving the scheduling problem. As shown in chapter 2, the previously used models do not accurately represent radio networks. By selecting any of these models a different problem is addressed.

In radio network scheduling the communication links between network transceivers are the essential information which must be used to determine a feasible schedule. As discussed in the chapter 2, there are certain characteristics which must be represented by the model, and others which can be subsumed under other, more important characteristics. For the purposes of scheduling, interference, antenna height, directional characteristics of the transmission pattern, etc., are all subsumed under the umbrella of transceiver range. This allows a problem definition in which the major characteristics of position and range are included.

Given a set of points, $p_i \in$ N–dimensional space, such that each point represents a location in the space, and a set of ranges $r_i \in R$, where each range r_i, is associated with a point p_i, a set of tuples is defined, $< p_i, r_i >$. A graph, G=(V,A), is constructed from this set of tuples. For every p_i, there is a vertex $v_i \in V$. If the Euclidean distance $d(p_i, p_j) \leq r_i$ then the directed arc $(v_i, v_j) \in A$. A graph constructed in this way from a set of points is called a *Point Graph.*

This definition allows the representation of networks in N–dimensions. In practice, only one, two, and possibly 3–dimensional *point graphs* are of current interest in modeling radio networks. Most networks are defined in 2–dimensional space, and only two dimensional test cases will be used in chapters 4 and 5. Given a network description of transceiver positions, t_i and ranges r_i, a point graph can be constructed: $G = (V, A)$, where $\forall t_i$ in the network, $\exists v_i \in V$ and $\forall t_i, t_j$ if the Euclidean distance $d(t_i, t_j) < r_i$ where r_i is the range of transceiver i, then $(v_i, v_j) \in A$. For the rest of this work v_i will be used in place of t_i in most formulae and definitions.

The term *Linear Point Graph* is used to indicate a point graph created from a set of points in 1–dimensional space. It has the property that the graph, G, representing the network can be drawn as a straight line. Some special purpose networks exhibit this property. It is not necessary that the transceivers actually form such a line, only that it be possible to represent their interconnections by a set of points on a line with the associated ranges. Similarly, the term *Planar Point Graph* will be used to define a point graph created from a set of points in 2–dimensional space (since 2–dimensions define plane). Most existing networks are accurately modeled by 2–dimensional point graphs.

An additional restriction meriting special terminology is the case of fixed or uniform range *point graphs*. For these graphs, $\forall t_i,\ r_i = d$, for some constant d. While this definition is nearly identical to Hale and Clark, Colbourn, and Johnson [30, 15] and their definitions of *unit disk graphs*, the definition of the conditions for edge existence differ. This difference is due to the definition of interference used in creating these graphs. Hale uses unit disk graphs in developing solutions to the frequency assignment problem. Edges in these graphs exist between vertices when the corresponding disks overlap. If the constant, d, is defined to be half the transmission range of the transceivers in the network, then Clark, Colbourn, and Johnson's definition of edge existence can be used to develop a digraph representation useful for network scheduling.

A class of graphs similar to unit disk graphs which has been studied in the literature is the *space* graph [43]. As with unit disk graphs, space graphs are defined in terms of a constant range. In addition, space graphs are defined as undirected graphs, and as such, represent a subclass of point graphs.

Point graphs are not arbitrary, even when restricted to a uniform range. The graphs shown in figure 2.6 are forbidden subgraphs of the set of uniform range 2–dimensional *point graphs*. A uniform range point graph has the property that $(v_i, v_j) \in A$ iff $(v_j, v_i) \in A$, and the term *symmetric* will be used to describe these point graphs (however, it is important to note that *symmetric* point graphs may be constructed from a set of points with non–uniform ranges). The following diagrams indicate the reasons why some of the graphs in figure 2.6 are forbidden subgraphs.

Figure 3.1(a) is a forbidden subgraph for uniform range *planar point graph*. Figure 3.1(b) demonstrates that there is no physical location for point "c" which would not add edges (b, c) and (c, d). Likewise, a transceiver network composed of uniform transceivers cannot have a set of five transceivers with the bidirectional communication links as depicted in figure 3.1(a).

Figure 3.2(a) is also a forbidden subgraph for uniform range *planar point-graphs*. Figure 3.2(b) demonstrates that there is no physical location for point "g" which would not add edges (f, g) and (b, g). Just as in figure 3.1, there is no way a transceiver

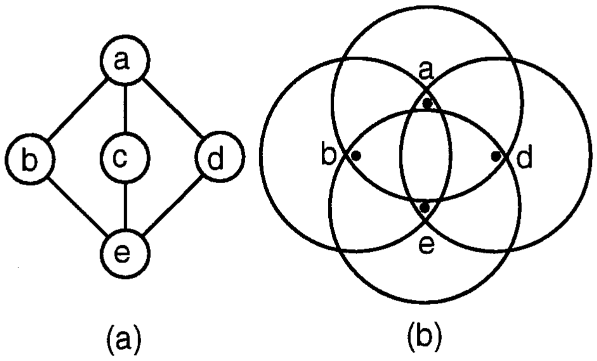

Figure 3.1: Justification of a forbidden subgraph for uniform range transceivers.

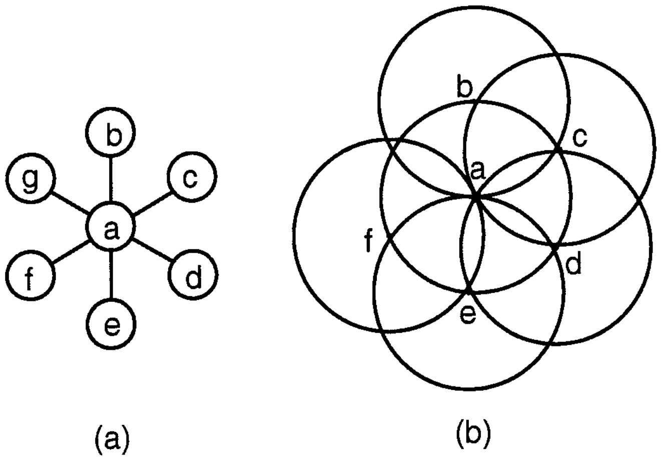

Figure 3.2: Another forbidden subgraph of uniform range point graphs.

network composed of uniform transceivers can have a set of transceivers with the bidirectional communication links as depicted in figure 3.2(a).

The remaining subgraphs in figure 2.6 have the same property in that there are no uniform transceiver networks which can have the communication links depicted. They all represent forbidden subgraphs (i.e., graphs which cannot appear as vertex induced subgraphs in any uniform range point graph.

An additional issue with respect to *point graphs* and forbidden subgraph characterizations, is that point graphs are defined as directed graphs, though symmetric point graphs are equivalent to undirected graphs. The forbidden subgraphs identified in figures 3.1 and 3.2 are only applicable for graphs generated from uniform range points, and which are therefore symmetric.

Forbidden subgraphs for non–uniform range point graphs are not as restricted since they do not necessarily result in symmetric point graphs. For example, if the undirected edges in figures 3.1 and 3.2 are replaced with the appropriate arcs, the vertex configuration shown in those figures can appear in point graphs.

Figure 3.3 shows non–symmetric point graphs which have the same vertex configuration as the forbidden subgraphs from figure 3.1 and figure 3.2.

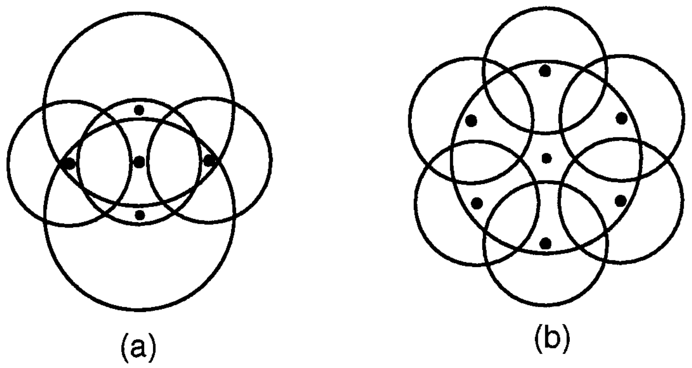

Figure 3.3: Non–uniform range point graphs with a set of points and ranges which correspond to the vertex configurations of figures 3.1 and 3.2.

Figure 3.3(a) is a non–uniform transceiver model similar to the forbidden subgraph in figure 3.1(a), while (b) shows a set of communication links similar to those in figure 3.2(a). The additional complication of directed graphs makes it necessary to define separate forbidden subgraphs for uniform and non–uniform range point graphs. One such graph is shown in figure 2.7.

Forbidden subgraph characterizations, while of interest to graph theoreticians, are outside the scope of this work. For this reason they will not be used to define point graphs in this dissertation. The question does arise as to exactly where point graphs fit in the set of all graphs.

Figure 3.4 shows a representation of the sets of graphs, and the relationship between trees, planar graphs, and point graphs.

In this figure, point graphs are also divided into *symmetric* and *asymmetric* graphs.

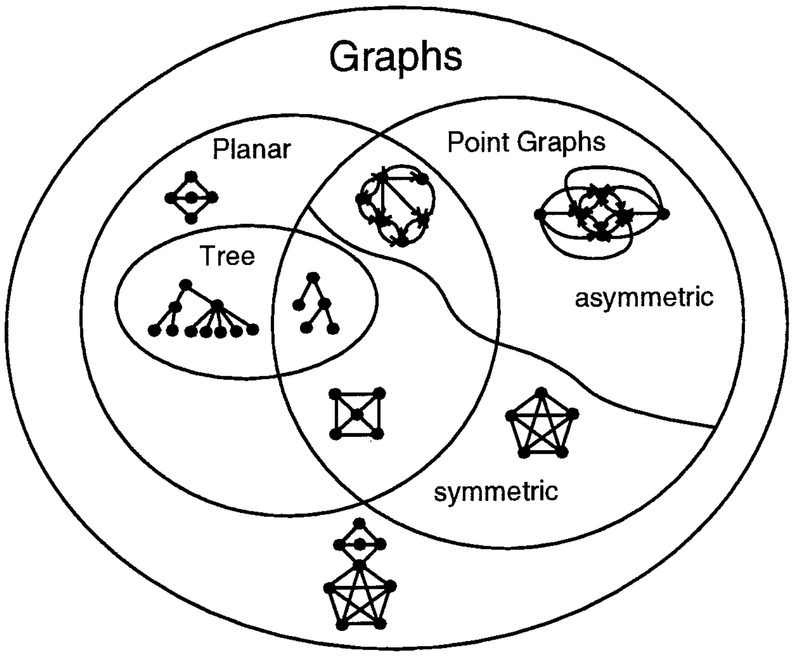

Figure 3.4: The set relationship between trees, planar graphs and point graphs.

As stated above, symmetric graphs are those for which $(v_i, v_j) \in A$ iff $(v_j, v_i) \in A$. Asymmetric point graphs are those for which this conditions does not hold.

Another relation between graph types is the relation between point graphs, unit disk graphs, and space graphs. This relationship is shown in figure 3.5.

As stated previously, unit disk graphs are 2–dimensional point graphs constructed from a set of points with a uniform range, r. The difference in definition between these graphs lies in the definition edges. For unit disk graphs, points within $2r$ of each other have an edge between them. Space graphs are defined for N–dimensional space, and any points separated by a Euclidean distance less than or equal to a constant. Both of these subsets of point graphs are symmetric graphs, as are all point graphs constructed from uniform range points.

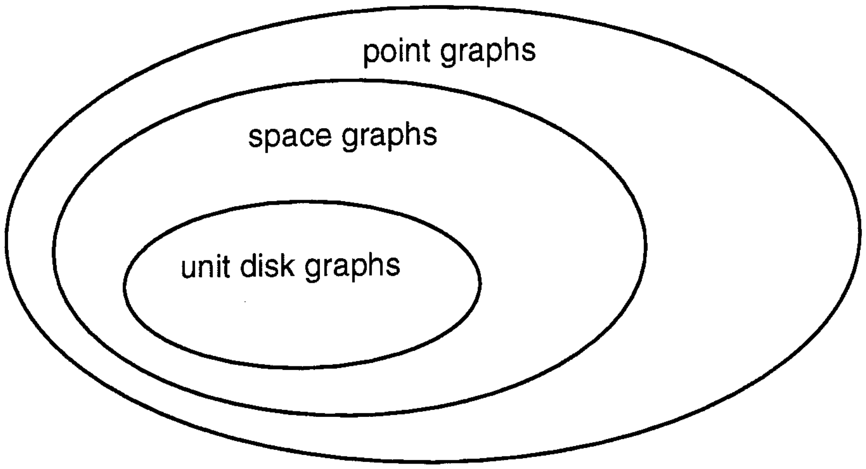

Figure 3.5: The set relationship between unit disk graphs, space graphs, and point graphs.

3.2 Directed Coloring

The term graph coloring is traditionally defined in terms of undirected graphs. A graph, $G = (V, E)$, is undirected if the edge set, E, is defined as a set of unordered pairs of vertices from the vertex set, V. Two vertices, $v_i, v_j \in V$ are said to be *adjacent* if there is an edge $(v_i, v_j) \in E$. Since the edge set contains unordered pairs, $(v_i, v_j) \equiv (v_j, v_i)$. Adjacency is also considered to be an irreflexive relation, therefore $\forall v_i \in V : (v_i, v_i) \notin E$.

3.2.1 Traditional Graph Coloring

Previous applications of graph coloring to network scheduling have relied on an undirected definition for the graph model [20, 30, 34, 48, 51]. The definition of coloring usually applied to graphs can be formally stated as follows:

> Given graph G = (V,E), a *distance-1 vertex coloring*, also called a *graph coloring*, is a mapping $\{\Phi : v \rightarrow \mathcal{N} | \phi(v_i) = \phi(v_j) \Leftrightarrow (v_i, v_j) \notin E\}$ where $v_i, v_j \in V$.

This is the traditional graph coloring problem, but as previously stated, network scheduling requires interference between transceivers to be considered. As a result, *distance–2 vertex coloring* has been defined to take secondary interference between transceivers into account. A formal definition of (undirected) distance–2 vertex coloring is:

> Given graph G = (V,E), a *distance–2 vertex coloring* is a mapping $\{\Phi : v \rightarrow \mathcal{N} | \phi(v_i) = \phi(v_j) \Leftrightarrow (v_i, v_j) \notin E$ and $\nexists v_k$ such that $(v_i, v_k), (v_j, v_k) \in E\}$ where $v_i, v_j, v_k \in V$.

Note that the undirected nature of G implies $(v_i, v_j) \equiv (v_j, v_i)$ (i.e., that edges are defined by unordered pairs).

3.2.2 Directed Coloring Defined

Directed versions of these same definitions permit more precise modeling of the networks being scheduled. The definitions parallel those for undirected graphs and can be stated as follows:

> Given digraph G = (V,A), a *directed distance–1 vertex coloring* is a mapping $\{\Phi : v \rightarrow \mathcal{N} | \phi(v_i) = \phi(v_j) \Leftrightarrow (v_i, v_j), (v_j, v_i) \notin A\}$ where $v_i, v_j \in V$.

The use of directed arcs necessitates the explicit mention of all the arcs. The unordered nature of edge definitions meant either permutation of the vertex pairs was sufficient to define edge existence. The definition for distance–2 coloring is similar:

> Given digraph G = (V,A), a *distance–2 vertex coloring* is a mapping $\{\Phi : v \rightarrow \mathcal{N} | \phi(v_i) = \phi(v_j) \Leftrightarrow (v_i, v_j), (v_j, v_i) \notin A$ and $\nexists v_k$ such that $(v_i, v_k), (v_j, v_k) \in A\}$ where $v_i, v_j, v_k \in V$.

Note that for secondary interference the only concern is incoming arcs to a third vertex, v_k. This is a significant issue in attempting to determine an optimal or near optimal network schedule using distance–2 vertex coloring.

3.2.3 Directed Coloring Justification

As stated in chapter 2, the use of undirected coloring introduces constraints on the coloring which do not necessarily exist in the network being modeled. These additional constraints, in the form of non–existent arcs, increase the number of colors used in coloring the graph, resulting in a longer schedule. The extent of this problem was investigated in [37], and is summarized here.

Four common greedy distance–2 coloring algorithms were used in comparing the efficacy of using a directed graph model rather than undirected in solving sample network scheduling problems. The first algorithm is a random, greedy coloring algorithm. Transceivers represented by the graph vertices are selected at random from among all vertices not yet colored and are colored as they are selected. This algorithm is represented by the column "Rnd" in table 3.1. The next algorithm used a maximum degree first heuristic for determining the order of vertices selected for coloring. As each node is selected for coloring its associated edges/arcs are removed and the degree of adjacent vertices are modified to maintain an accurate degree representation. The results of this algorithm are found in the columns labeled "Max" in the tables of results. The third algorithm used a minimum degree last heuristic for ordering vertices for coloring. As with the maximum degree heuristic, as a vertex is selected its adjacent vertices degrees are modified. These results are contained in the columns labeled "Min". Finally, a distance–2 minimum degree last heuristic was used to order the vertices. In this case, vertices were ordered according to the number of adjacent vertices and those at distance–2. These results are contained in the columns labeled "D–2".

Each table entry represents an average over 30 test cases of uniformly distributed transceivers. All transceivers were randomly positioned on a 200–by–200 grid, and assigned a range varying from 50-100% of the value found in the "Range" column in tables. Each such network was then colored using both undirected and directed graph representations.

For the sample data sets, directed coloring used 79% of the colors of the corresponding undirected coloring algorithms. The improvement depends on the density of the graph and the corresponding number of induced communication links described above. There are no circumstances under which an algorithm using the directed coloring definition will use more colors than the same algorithm using the traditional definition of graph coloring.

Table 3.1: Comparison of undirected and directed graph coloring algorithms.

No. Nodes	Range	Undirected				Directed			
		Rnd	Max	Min	D–2	Rnd	Max	Min	D–2
200	20	12.40	12.40	11.47	11.43	10.33	11.03	9.20	9.43
	30	22.86	22.63	21.16	21.13	18.83	17.76	16.76	16.46
	40	36.73	35.56	33.80	33.56	28.83	27.40	25.56	25.63
250	20	15.93	15.80	14.86	14.70	13.30	12.73	11.83	11.86
	30	28.50	27.66	25.96	25.76	22.50	22.06	20.23	20.30
	40	45.86	44.23	42.30	42.10	35.40	33.60	32.03	31.43
300	20	17.96	17.56	16.66	16.50	14.73	14.33	13.36	13.20
	30	33.73	32.33	31.20	30.93	26.13	25.00	23.90	23.30
	40	53.76	52.43	49.33	48.83	41.93	39.96	37.43	36.93

Due to the disparity in performance cited above, directed coloring will be used for all further work in this dissertation, unless stated otherwise.

In this chapter, a new graph type was defined for modeling radio networks. This model accurately represents the communications between transceivers in the network. In addition, the problem of directed coloring was defined and shown to be superior to the traditional model when dealing with asymmetric graphs.

Chapter 4

Broadcast Scheduling

This chapter examines broadcast scheduling of radio networks. The first section looks at the complexity of scheduling by analyzing vertex coloring of point graphs. The next section discusses the development of approximation algorithms based on geometric properties of point graphs. The performance of these algorithms are compared to other coloring algorithms in the next section. The final section discusses the results obtained in the preceding sections.

4.1 Complexity of Coloring Point Graphs

This section explores the complexity finding an optimal solution to the problem of vertex coloring point graphs. As mentioned previously, optimality is of interest due to the cumulative effect of non–optimal schedules (a 1% error is approximately 15 minutes per day). Since previous work has focused on undirected coloring, both undirected and directed coloring are examined, and algorithms or proofs are given to support arguments about the complexity of each case. To provide structure, this section first explores undirected coloring of 1–dimensional and 2–dimensional point graphs. The section concludes by examining directed coloring for both types of graphs.

4.1.1 Undirected Coloring

As previously stated, undirected coloring is the "traditional" coloring problem. Point graphs are inherently directed, and while it may not make sense to use the concept of undirected coloring for them, the common practice has been to treat directed graphs as undirected when coloring them. However, in the special case of uniform range point graphs undirected and directed coloring are identical. It is also true that in the case of distance–1 coloring, the distinction between directed and undirected coloring disappears. So while it may be irrelevant to the application area of radio networks, undirected coloring of point graphs is discussed for the sake of completeness.

Distance–1 Coloring of Linear Point Graphs

Undirected linear point graphs are a special case of graph coloring which can be solved in polynomial time. Some examples of this type of graph is shown in figure 4.1. (Recall that the name "linear" refers to the positions of the points not to the connections between points.)

This type of problem is similar to the channel–assignment problem in VLSI layout and design. An optimal polynomial time algorithm for the channel–assignment problem was proposed by Gupta, Lee and Leung [28]. While their algorithm was intended for channel–assignment, it can be used to color an interval representation of a linear point graph modeling a network of uniform range transceivers. The input to the algorithm is a set of intervals, where each interval corresponds to half the range of each transceiver. For each transceiver, i, let x_i represent half the range in one direction of the linear network, called a *left* endpoint, and y_i represent half the range in the other direction, a *right* endpoint. These endpoints define a set of intervals where any two overlapping intervals cannot have the same color (or channel assignment).

Gupta, Lee, and Leung's algorithm is as follows:

```
Step 1: Sort the 2N endpoints so $z_i \leq z_{i+1} \ldots z_{2N}$.[1]
Step 2: $COUNTER \leftarrow 0$, $MAXCOLOR \leftarrow 0$
    $COLOR \leftarrow 1$
Step 3: for i = 1 to N do:
```

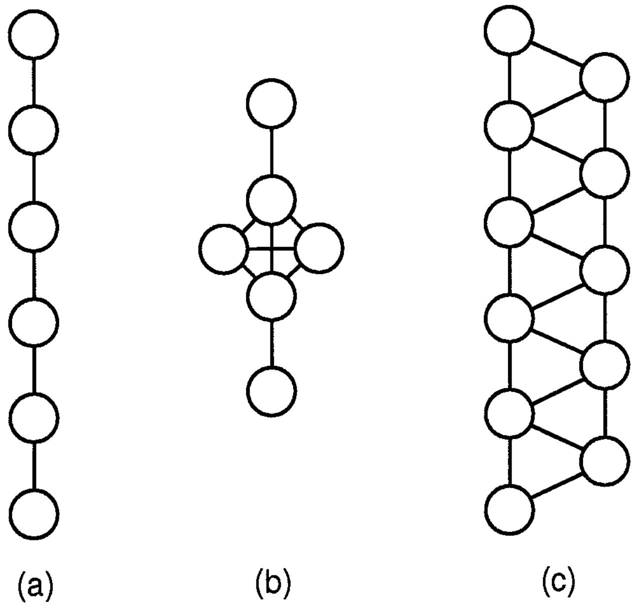

Figure 4.1: Some simple examples of uniform range linear point graphs.

```
    $NEXT(i) \leftarrow i+1$
Step 4: for j = 1 to 2N do:
    if $z_j$ is a left endpoint, $x_k$
        $COUNTER \leftarrow COUNTER + 1$
        $MAXCOLOR \leftarrow max\{COUNTER, MAXCOLOR\}$
        $ASGN(k) \leftarrow COLOR$
        $COLOR \leftarrow NEXT(COLOR)$
    else $z_j$ is a right endpoint $y_k$
        $COUNTER \leftarrow COUNTER - 1$
        $TEMP \leftarrow ASGN(k)$
        $NEXT(TEMP) \leftarrow COLOR$
        $COLOR \leftarrow TEMP$
```

An argument for the correctness of this algorithm is presented in [28] and will not be presented here. The algorithm itself requires $O(n \ln n)$ time to execute. For graphs where each point may have a different range, this algorithm must be modified. Figures such as figure 4.2 can be distance–1 colored by using the following modification to Gupta, Lee, and Leung's algorithm.

[1]If $z_i = z_{i+1} \ldots = z_j$, the right endpoints appear before the left endpoints.

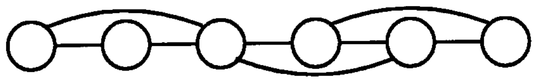

Figure 4.2: An undirected representation of a non-uniform range linear point graph.

Note that in cases where variable ranges occur, the simple interval algorithm above will not work because it relies on a technique of using half the range determining the values of the left and right endpoints. While two transceivers are in range of each other when their left and right endpoint are at the same position and their ranges are identical, no such statement can be made if the ranges vary.

First, the modified algorithm uses four points for each transceiver. As with the preceding algorithm left and right endpoints are defined, however these are based on the full range rather than half the range. In addition, the other points, lp_i and rp_i, representing the position of transceiver i are added, and are called the left position and right position points, respectively. The algorithm then becomes :

```
Step 1: Sort the 4N endpoints so $z_i \leq z_{i+1} \ldots z_{4N}$.[2]
Step 2: $COUNTER \leftarrow 0$, $MAXCOLOR \leftarrow 0$
    $COLOR \leftarrow 1$
Step 3: for i = 1 to N do:
    $NEXT(i) \leftarrow i + 1$
    $ACTIVE(i) \leftarrow FALSE$
    for j = 1 to N do:
        $INRNG(i, j) \leftarrow FALSE$
Step 4: for j = 1 to 4N do:
    if $z_j$ is a left endpoint, $x_k$
        for i = 1 to $N$ do:
            $CLIST(k, i) \leftarrow NEXT(i)$
        $LCLR(k) \leftarrow COLOR$
        $ACTIVE(k) \leftarrow TRUE$
    else if $z_j$ is a right endpoint, $y_k$
        $COUNTER \leftarrow COUNTER - 1$
        $TEMP \leftarrow ASGN(k)$
        $NEXT(TEMP) \leftarrow COLOR$
        $COLOR \leftarrow TEMP$
        for i = 1 to N do:
            if $ACTIVE(i) \wedge \neg INRNG(k, i)$
            $TEMP \leftarrow ASGN(k)$
            $CLIST(i,TEMP) \leftarrow LCLR(i)$
            $LCLR(i) \leftarrow TEMP$
else if $z_j$ is a left position, $lp_k$
    $COUNTER \leftarrow COUNTER + 1$
    $MAXCOLOR \leftarrow max\{LCLR(k), MAXCOLOR\}$
    $ASGN(k) \leftarrow LCLR(k)$
    if $COLOR \neq LCLR(k)$
        $TEMP \leftarrow COLOR$
        while $NEXT(TEMP) \neq LCLR(k)$ do:
            $TEMP \leftarrow NEXT(TEMP)$
        $NEXT(TEMP) \leftarrow NEXT(LCLR(k))$
    else
        $COLOR \leftarrow NEXT(COLOR)$
    $ACTIVE(k) \leftarrow FALSE$
    for i = 1 to N do:
        if ACTIVE(i) $\wedge$ $LCLR(i) \neq LCLR(k)$ then
            $TEMP \leftarrow LCLR(i)$
            while $CLIST(i,TEMP) \neq LCLR(k) \wedge$
                    $TEMP < MAXCOLOR$ do:
                $TEMP \leftarrow CLIST(i,TEMP)$
            if $TEMP < MAXCOLOR$ then
                $NEXT(TEMP) \leftarrow NEXT(LCLR(k))$
        else if ACTIVE(i)
            $LCLR(i) \leftarrow CLIST(i, LCLR(i))$
else $z_j$ is a right position, $rp_k$
    for i = 1 to N do:
        if ACTIVE(i) then
            $INRNG(k,i) \leftarrow TRUE$
```

In this algorithm, a separate list is maintained for each transceiver to keep track of other transceivers whose positions are in range of the transceiver being colored. The disparity in ranges makes it necessary to maintain this information to determine an optimal coloring. For example, in figure 4.3 the intervals for transceiver 1 and 3, 2 and 4, 3 and 4, and 2 and 5 overlap, but there is no communication link between of these transceiver pairs.

[2]If $z_i = z_{i+1} = \ldots = z_j$, the left position points appear before right endpoints which appear before the left endpoints which appear before right position points.

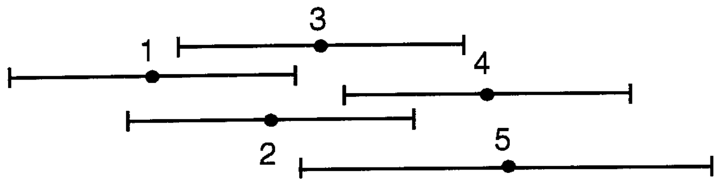

Figure 4.3: Interval representation of an non–uniform range linear point graph.

As the left endpoint (extreme range) for a transceiver is encountered, the current list of available colors is started for that transceiver. This list is maintained separately for each transceiver. As the right endpoint is reached for other transceivers, the entry in array INRNG indicates if the transceiver position is within the range of any other not yet colored transceiver. If it is not in range, the transceiver whose right endpoint has been reached can have its color returned to the list of available colors for the not yet colored transceiver, otherwise the color of that transceiver remains unavailable for the not yet colored transceiver and its list of available colors remains unmodified.

Left and right position points allow a transceiver vertex to be colored based only on those transceivers whose positions are to the left of the one being colored, while at the same time maintaining information about which transceivers to the right have the one being colored within their range. As a transceiver is colored, that color must be removed from the list of available colors for all those nodes not yet colored, providing the color appears in the list. Figure 4.4 shows a situation where the assigned color may not appear in the list of available colors.

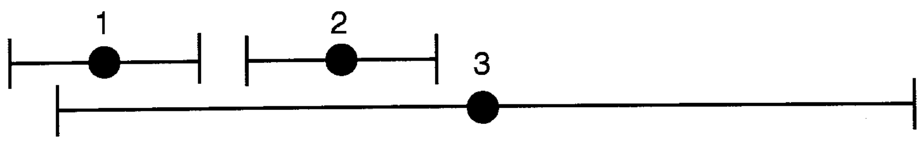

Figure 4.4: Interval conditions under which a color assigned to transceiver 2 will not be in the list of available colors for transceiver 3.

Assume transceiver 2 is assigned the same color as transceiver 1. As transceiver 2 is colored, its color will not appear in the list of colors for transceiver 3, and will not need to be removed from the 3's list of available colors because transceiver 1 is within the range of 3. Therefore as the right endpoint of 1 is reached, the list of available colors for 3 will not be changed (this is also true when the right endpoint of transceiver 2 is reached).

Maintaining these lists increases the complexity of the overall algorithm from $O(n \ln n)$ to $O(n^2)$, though it remains polynomial. The assignment of a color is postponed until the position of the transceiver is reached to prevent erroneous determination of interference between transceivers which only share overlapping transmission ranges.

Distance–2 Coloring of Linear Point Graphs

The optimal algorithm for distance–2 coloring presented here is also based on Gupta, Lee, and Leung's algorithm. Three values are generated for each transceiver in the linear network, one each for a left endpoint, a right endpoint, and the position of the transceiver.

```
Step 1: Sort the 3N endpoints so $z_i \leq z_{i+1} \ldots z_{3N}$.[3]
Step 2: $COUNTER \leftarrow 0$, $MAXCOLOR \leftarrow 0$
    $COLOR \leftarrow 1$, $NUMTOPROC \leftarrow 0$
Step 3: for i = 1 to N do:
    $NEXT(i) \leftarrow i + 1$
Step 4: for j = 1 to 3N do:
    if $z_j$ is a left endpoint, $x_k$
        $NUMTOPROC \leftarrow NUMTOPROC + 1$
        $WAITING(NUMTOPROC) \leftarrow k$
    else if $z_j$ is a right endpoint $y_k$
        $COUNTER \leftarrow COUNTER - 1$
        $TEMP \leftarrow ASGN(k)$
        $NEXT(TEMP) \leftarrow COLOR$
        $COLOR \leftarrow TEMP$
    else $z_j$ is a position point $p_k$
        for i = 1 to NUMTOPROC do:
            $COUNTER \leftarrow COUNTER + 1$
            $MAXCOLOR \leftarrow max\{COUNTER,$
                        $MAXCOLOR\}$
            $ASGN(WAITING(i)) \leftarrow COLOR$
            $COLOR \leftarrow NEXT(COLOR)$
```

$NUMTOPROC \leftarrow 0$

The following argument shows the correctness of this algorithm.

Given an linear point graph with N vertices. The first step of the algorithm is to construct a set of points such that $x_i \leq p_j \leq y_i$ iff arc $(v_i, v_j) \in A$. Note that $\forall_i\, 1 \leq i \leq N$, $x_i \leq p_i \leq y_i$. These points are sorted by their location to produce a list, Z. Ties between points in the sorting are resolved according to the type of point. If $x_i = p_j = y_k$ for transceivers i,j, and k, the points will be ordered in the sort such that if $z_l = x_i$, $z_m = p_j$, and $z_n = y_k$ then $l < m < n$.

At all times during the execution of this algorithm, NEXT is a stack of the colors which are available, and WAITING is a list of vertices for which the left endpoint, x_i has been reached but for which no color has been assigned. When a position point, p_j is reached colors are assigned to those vertices which do not already have a color. The following conditions exist for all vertices:

- **Case 1:** x_i has not been encountered. This vertex does not affect the colors available to color p_j in a greedy manner and can be ignored for the moment.

- **Case 2:** y_i has already been encountered. This vertex is not in range of p_j, and since j has not been colored, v_i is not in the range of p_j

- **Case 3:** v_i has a color, and y_i has not been encountered. v_j is distance–1 from this vertex and must have a different color. In addition, all other vertices distance–1 from v_j must have a different color from v_i (the distance–2 requirement).

- **Case 4:** x_i has been encountered, but p_i has not, and v_i has not been assigned a color. This vertex is one of those contained in the list WAITING. It is distance–1 from v_j, and distance–2 (and distance–1) from all other vertices in the WAITING list and those meeting condition 3 (which are all other vertices which are currently assigned a color).

[3]If $z_i = z_{i+1} = \ldots = z_j$, the left endpoints appear before position points which appear before the right endpoints.

As the algorithm proceeds, at each position point, colors are assigned only to those vertices meeting condition 4. At each such point the total number of colors used equals the degree of the node plus one, which is optimal for distance–2 coloring. All colors which are not assigned to a vertex meeting condition 3 are on the stack NEXT, guaranteeing an optimal solution to coloring for a undirected linear Point graph.

The time complexity for this algorithm is determined by the time to sort the points, and is $O(n \ln n)$.

Distance–1 Coloring of Planar Point Graphs

The coloring problem for arbitrary graphs has long been known to be an NP–complete problem. In fact, Garey and Johnson show that it is likely that no good approximation algorithms may be found (a "good" approximation algorithm is defined as one whose performance is bound by a constant, that is, its answer is always $\leq c * optimum$) [23]. The planar point graph problem has been investigated by various authors for the uniform range case called unit disk graphs [11, 15, 30]. The previous work described this problem as partitioning points in the plane by Burr [11], and distance–1 coloring of unit disk graphs by Hale and by Clark, Colbourn and Johnson [30, 15]. This problem has been shown to be NP–complete with Burr providing a proof which is restated using a different construction technique by Clark, Colbourn, and Johnson.

The NP–complete nature of this problem leads to the common use of approximation algorithms for finding reasonable rather than optimal solutions. The interested reader is referred to Burr for a geometric proof. We describe the proof discussed by Clark et. al.

The proof is based on a polynomial time transformation from PLANAR GRAPH 3-COLORABILITY with maximum degree 3 (shown to be NP–complete in [24]). They transform a planar graph, G = (V,E), with maximum degree 3 into a unit disk graph, $G\prime$, such that G is 3–colorable iff $G\prime$ is 3–colorable. Their transformation uses a result by Valiant[57]:

A planar graph with maximum degree 4 can be embedded in the plane using $O(|V|)$ area in such a way that its vertices are at integer coordinates and its edges are drawn so that they are made up of line segments of the form $x=i$ or $y=j$, for integers i and j.

Each edge between nodes u and v, for all u and v, is then replaced by a construct such as in figure 4.5, which can be bent around corners provided the segments are at least 10 units long. This requirement can be ensured by the construction.

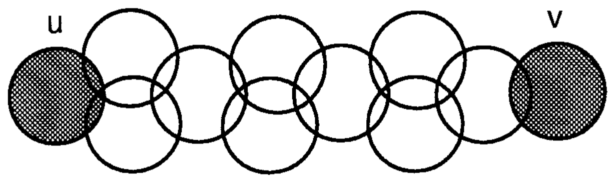

Figure 4.5: Transformation construct for an edge between vertices u and v, used in NP–completeness proof by Clark, Colbourn, and Johnson.

An examination of this construction will show that in a three coloring of the graph $G\prime$, vertices u and v must be assigned colors which are different from each other. Therefore, if $G\prime$ is 3–colorable, must G also be 3–colorable.

The NP–completeness for distance–1 coloring of planar point graphs follows from the NP–completeness results for the subset, unit disk graphs.

Distance–2 Coloring of Planar Point Graphs

There are various approaches to proving distance-2 vertex coloring NP–complete. For planar graphs, Ramanathan and Lloyd [46] proved the 7–planar–distance–2 coloring problem is NP–complete by construction from a 7–distance–2 coloring proof for arbitrary graphs in [21] (Even et.al. proved a related problem by component design, and argued that modification of their construction could be used for k–distance–2 coloring of arbitrary graphs. The choice of k=7 for Ramanathan and Lloyd is determined by the structure of their component constructs used in converting arbitrary graphs to planar graphs). Hu uses a technique first proposed by Ramaswami in proving NP–completeness [33]. A later more elegant version of Ramaswami's proof for arbitrary graphs appears in [51, 20]. Of these proofs, the most general is the one from [51, 20]. Unfortunately, the results for arbitrary and planar graphs do not provide any information about planar point graphs. As demonstrated earlier, planar point graphs are neither arbitrary nor are they planar, therefore results for these two classes of graph cannot be extended to planar point graphs.

Though they discuss undirected distance–1 coloring, Clark, Colbourn, and Johnson do not address undirected distance–2 coloring of unit disk graphs.

A proof of the NP–completeness of uniform range planar point graph 7–colorability can be constructed using a result from [8], and component constructions similar to those in [48]. The result from Biedl and Kant [8] can be stated as follows:

Lemma 4.1 *Given a connected graph, G, the vertices and edges can be embedded in a plane in linear time.*

This lemma, and the components shown in [48] allow an instance of uniform range planar point graphs to be produced by transforming a connected graph. This instance is then distance–2 7–colorable iff the original connected graph is 7–colorable. Graph 7–colorability is NP–complete since undirected connected graph k–coloring is NP–complete for all fixed $k \geq 3$ [23]. Tighter bounds on this may be possible.

Distance–1 Coloring of *D*–Dimensional Point Graphs

The colorability of D–dimension point graphs for $D > 2$ is NP–complete as a direct result of the NP–completeness of undirected coloring for planar point graphs, a subset of D–dimension graphs.

Distance–2 Coloring of *D*–Dimensional Point Graphs

D–dimensional point graph colorability is NP–complete. This is a direct result of the NP–completeness of 2–dimensional point graphs.

4.1.2 Directed Coloring

Distance–1 Coloring of Linear Point Graphs

The algorithm presented for undirected distance–1 coloring of linear point graphs will also produce an optimal, conflict free directed distance–1 coloring. The reason for this is that the criterion for distance–1 directed coloring relies only on the existence of an arc between the nodes being colored, not on the direction of the arc. This problem can be optimally solved in $O(n \log n)$ time using the algorithm previously described.

Distance–2 Coloring of Linear Point Graphs

This problem can also be optimally solved in polynomial time. The algorithm described for undirected coloring will also optimally determine a directed coloring. An examination of this algorithm shows that the points are colored in order of their left endpoints. The optimality of the algorithm can be shown by examining the coloring process. First note the minimum number of colors necessary to distance–2 color a graph must be at least the maximum in–degree of all the vertices in the graph plus one. The other characteristic to note is that the array NEXT acts like a stack which contains all possible colors. As the algorithm proceeds, the contents of the stack are "popped" off as a vertex is colored, and "pushed" back on the stack when a vertex no longer affects the coloring of other vertices.

Figure 4.6 shows a sample point i and the arcs to and from that point. When the position point for i is reached, the color assigned to vertex n will be the one which was last placed on NEXT. When co–located points are sorted, the left endpoints will precede the position points, which will precede the right endpoints. Therefore, WAITING will contain i, p, and q, in that order even though the left endpoint for q is co–located with the position point for i. Vertex i will then receive the same color as vertex n (the one on the "top" of the stack), and p and q will receive colors different from vertices m and o due to the secondary interference between them.

After i, p, and q are colored there are a total of 5 of the N original colors contained in NEXT which are currently off the stack. This is exactly the degree of i plus one. The colors assigned to o and m are then placed on NEXT as their right endpoints are

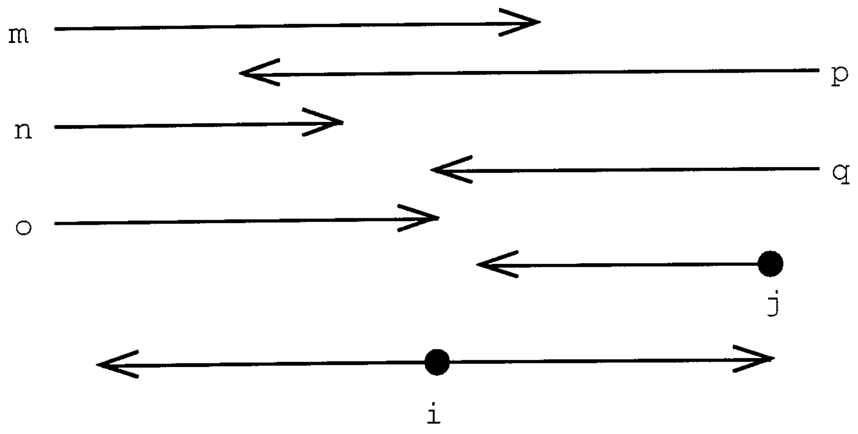

Figure 4.6: A sample linear point graph node and the arcs to and from it.

reached. When the position point for j is reached it is the only vertex in the array WAITING, and is assigned the same color as m. Because this is a directed coloring j and m do not have secondary interference and may therefore have the same color. It should be clear from this discussion that this algorithm remains optimal for directed coloring, and that distance–2 directed coloring of linear point graphs can be solved in $O(n \ln n)$.

Distance–1 Coloring of Planar Point Graphs

This problem is NP–complete as a direct result of the NP–completeness of unit disk graphs [11, 15]. The reason for this correspondence is twofold. First, the definition of directed coloring makes arc existence the sole requirement for adjacency. This has the result that for distance–1 coloring undirected and directed coloring are identical. Second, since the special case of uniform range (unit disk) planar point graph distance–1 undirected coloring is NP–complete, distance–1 directed coloring is also NP–complete.

Distance–2 Coloring of Planar Point Graphs

The following is a proof of the NP–completeness of planar point graph distance–2 colorability.

Theorem 4.1 *2-Dimensional Point Graph Distance–2 3–Colorability (also called Planar Point Graph Distance–2 3–Colorability, PPGD2) is NP–Complete.*

Proof: It can easily be shown that PPGD2 $\in$ NP, a nondeterministic algorithm can be used to assign a coloring, the graph can then be checked in polynomial time to determine that vertices u and v have different colors if the Euclidean distance $d(u, v)$ is less than the ranges of either u or v, or if $\exists w \in$ V such that $d(u, w) <$ range of u and $d(v, w) <$ range of v.

The proof is a proof by a component design construction which transforms the problem, *Planar Graph 3-Colorability with Node Degree At Most 4* (shown NP–Complete by Garey, Johnson, and Stockmeyer [24]), into PPGD2. As previously noted, Biedl and Kant [8] present a linear time algorithm which produces an embedding of graph G=(V,E) on a grid such that each vertex is at a grid point and each edge is composed of vertical and horizontal segments with at most two bends. Start with an arbitrary instance, G=(V,E), of a planar graph with node degree at most 4, use Biedl's algorithm to produce an embedding in the grid scaled by a factor of 3 (guarantees a distance of at least three between all points).

The remainder of the construction replaces the embedding with points in the 2–dimensional grid. There are two categories of points in the construction. Vertex replacement points replace the embedded vertices, and edge replacement points are positioned along the horizontal and vertical segments of the embedded edges. Edge replacement points are further divided into two types: biconnected points, and singly connected points. In all, three types of replacement points will be used. Type–1, vertex replacement points have a range of one unit, type–2, bidirectional edge replacement points have a range of 0.25 units, and type–3, singly connected edge replacement points have a range of 0.45 units.

Type–1 vertex replacement points are positioned at the scaled embedding points for the vertices of the planar graph. Each vertex replacement point is assigned a range of one unit. The result is a set of such points, P, such that each $p_i \in P$, corresponds to a vertex from G, with position x_i, y_i from the embedding and range $r_i = 1$ unit.

Edge replacements are constructed by visiting each vertex replacement point in breadth first order. For each vertex all edges not yet replaced are replaced by determining positions for edge replacement points which follow the segments of the embedded edges. The positions for edge replacement points are shown in figure 4.7. The coordinates for the positions are expressed with respect to the vertex replacement point, v, located at (0,0), and with first segment of the edge being replaced as the positive x–axis. (The values used in this construction are not the only one which could be used, but are provided as a definition of exact values which create the desired properties in the resulting point graph.)

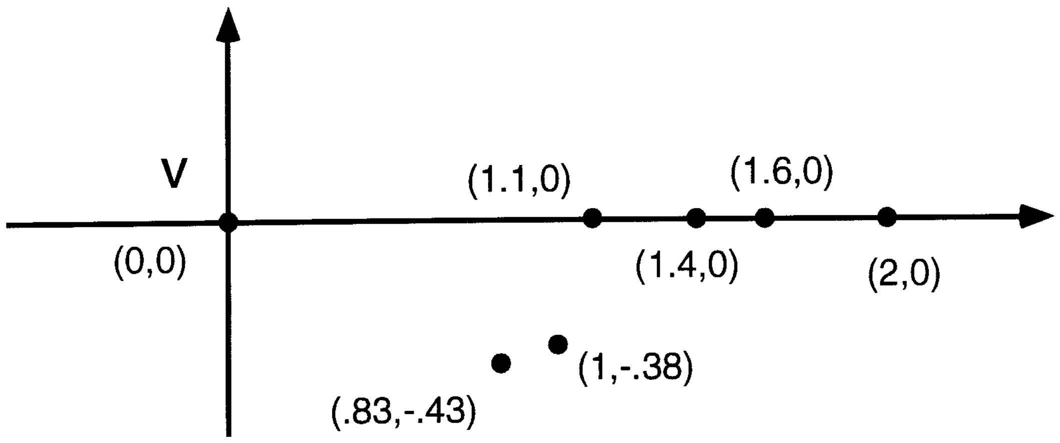

Figure 4.7: Edge replacement point positions relative to the start vertex v and the first segment of the edge being replaced.

Type–2 biconnected edge replacement points are used at coordinates (0.83, -0.43) and (1, -0.38) with a type–3 singly connected point at (1.1,0). This pattern is repeated with two more type–2 points at (1.4,0) and (1.6,0) followed by a type–3 point located at (2,0). For all but the last unit of the segments of the embedded edges, two type–2 and one type–3 point are positioned along the segment of the edge being replaced such that they are 0.4, 0.6, and 1.0 units distance from the last type–3 point whose position was determined. The final unit distance of the last segment of the embedded edge is effectively replaced by the last type–3 point in the construct. Figure 4.8 shows the edge replacement for an edge between two vertex replacement points 3 units apart in the scaled embedding.

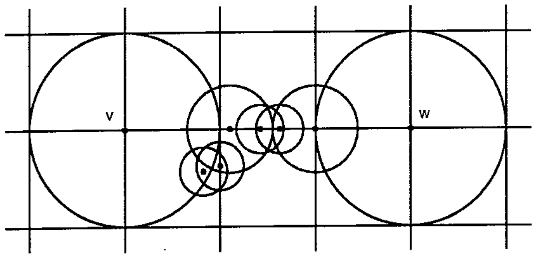

Figure 4.8: Complete edge replacement between vertex replacement points v and w, 3 units apart in the scaled embedding.

As figure 4.8 and figure 4.7 show, the combination of positions and ranges require the scaling factor of 3. Each type–3 point in the edge replacement, as shown in figure 4.9, will have the same color as vertex replacement point v, in a distance–2 3–coloring of the resulting PPGD2.

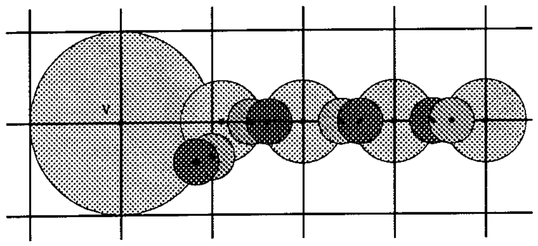

Figure 4.9: 3–coloring of vertex replacement point, v, and an edge replacement starting at that point. The type–1 and type–3 points are the same color.

Each edge replacement has the property that the resulting 2–dimensional point graph has an arc pattern similar to the one shown in figure 4.10.

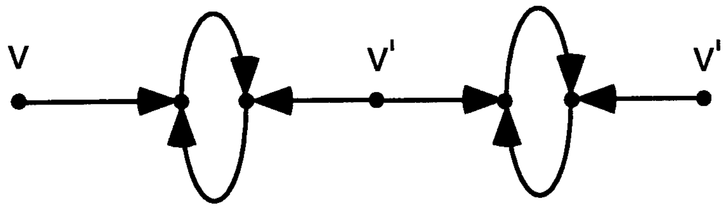

Figure 4.10: Directed graph arc pattern introduced by edge replacement.

To show that this is a viable replacement process two properties must be maintained. First it must be possible to make a "bend" in the edge without changing the arc pattern shown in figure 4.10 (recall Biedl and Kant showed each edge replacement will have at most two bends). A "bend" in the embedded edge is a 90° turn, a change from vertical to horizontal or horizontal to vertical trace.

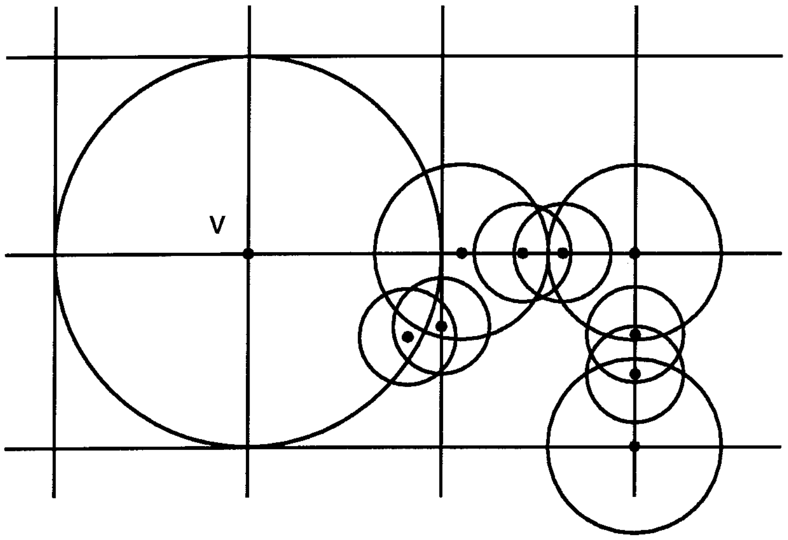

Figure 4.11: An edge construct with a 90 degree bend, showing that such bends are possible without causing interference

Due to the scaling factor of 3, bends will occur at multiples of 3 units from either the starting vertex replacement point for the edge or from the last bend in the edge. For diagramming simplicity this property was violated in figure 4.11.

The second property is that the edge constructs must allow a vertex, v from G to have four non-interfering edge replacements. Figure 4.12 shows one such node. Each type–3 edge replacement point, labeled v', will have the same color as vertex v in a distance–2 3–coloring of the generated instance of PPGD2.

Each vertex replacement point is visited in a breadth first order and each edge not yet replaced, is replaced as described above. This process continues until all type–1 vertices (and therefore all $v \in V$ from the original graph G) have been visited. For each embedded edge with embedded segments of total length s, there are $3s - 1$ points added by the construction process. Let m be the maximum length of any edge

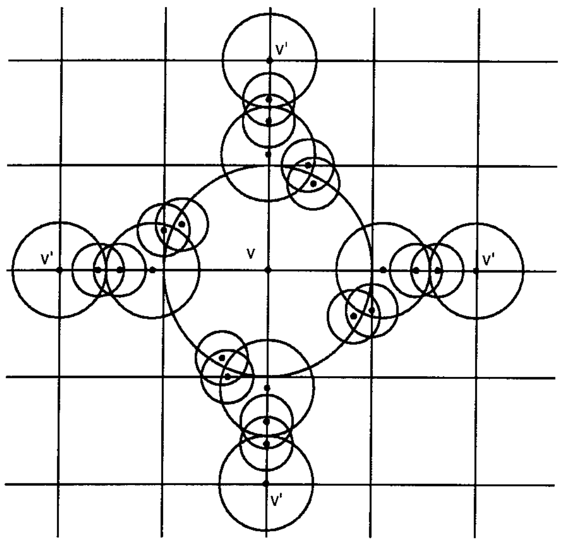

Figure 4.12: The construction for a vertex, v, of degree 4 from the original planar graph G. Each vertex, v' has the same color as vertex v in a valid distance–2 3–coloring of the constructed graph, G'.

embedding, there are then $O(|V| + 3m|E|)$ points in the 2–dimensional point graph produced by the process of transforming the problem to an instance of PPGD2. The total time complexity to produce the instance of PPGD2 is $O(|V|)$ (based on the linear time algorithm from [8] to produce the embedding, the degree restriction to at most 4, and the area requirement results of [57]).

First show that if G is 3–colorable, PPGD2 is distance–2 3–colorable. As shown by the arc representation in figure 4.10, any vertex replacement point combined with its edge replacement points is distance–2 3–colorable. Then for any two adjacent nodes, v and w in G, we can say without loss of generality that the vertex replacement for w will be "adjacent" to a type–3 point of the same color as v. Since this type–3 point will have only one incoming arc from the vertex replacement point for w, the only coloring restriction is that they not have the same color, which is equivalent to v and w not having the same color in a distance–1 coloring in the original problem. Therefore, if there is a distance–1 3–coloring for the instance G, of planar graph with node degree at most 4, the generated instance of PPGD2 is distance–2 3–colorable.

For the reverse, assume there is a distance–2 3–coloring of the instance of PPGD2. This means that for all vertex replacement points, v and w, if v and w have the same color, there is no edge replacement between them. If such an edge replacement existed, some type–1 vertex replacement point w, would have a type–3 edge replacement point, v', positioned such that the Euclidean distance, d(w,v') $\leq r_w$, and points w and v' would have the same color contradicting our assumption of a distance–2 3–coloring. Therefore, PPGD2 is NP–Complete. □

Distance–1 Coloring of *D*–Dimensional Point Graphs

The colorability of D–dimension point graphs for $D > 2$ is NP–complete as a direct result of the NP–completeness of directed coloring for planar point graphs, a subset of D–dimension graphs.

Distance–2 Coloring of *D*–Dimensional Point Graphs

D–dimensional point graph colorability is NP–complete. This is a direct result of the NP–completeness of 2–dimensional point graphs.

4.2 Algorithm Development

The complexity results of the previous section indicate that approximation algorithms will be the most efficient means of determining conflict free schedules. Two types of algorithms are developed in this work. The first type depend on the geometry of the problem as determined by the locations of the transceivers in the network. The second type rely only on the network connection topology to determine coloring order. In all cases the color is greedily assigned .

4.2.1 Geometric Algorithms

Distance–2 coloring of linear point graphs can be optimally solved in polynomial time. This observation is the basis for the first of the geometric algorithms. The other two geometric algorithms partition the area occupied by the transceivers in order to determine the order of coloring used. Graph coloring algorithms typically address the topology of the graph. Because point graphs are defined in terms of geometric locations from the problem domain, this information can be used in coloring point graphs for which this geometric carried over from the problem domain. The effect of geometry on the coloring process has never been explored. These geometric algorithms are unique in their approach to the problem of graph coloring.

Linear Projection

The first algorithm uses the linear point graph distance–2 directed coloring algorithm to color a 2–dimensional network. In order to do this a 1–dimensional network must be produced. The problem with a non-linear network is determining a good projection line for the problem. A good line will induce the fewest number of arcs (edges) in the projection. For example, given the network of 4 uniform range transceivers shown in figure 4.13, we could color this network with one color (for both distance-1 and distance-2 colorings).

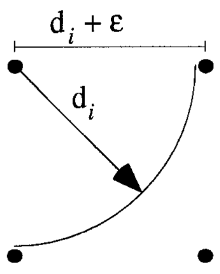

Figure 4.13: A simple uniform transceiver network.

Since information is lost in projecting 2–dimensional information to one dimension, any resulting coloring (schedule) may not be optimal (though this will depend on the structure of the network being modeled). As shown in figure 4.14, a projection to either line A or B will result in a 2–coloring of the graph (for both distance-1 and distance-2). A worse selection for projection line would be line C, which is 3–distance-1 colorable and 4–distance-2 colorable.

It is desirable to select the best projection line for the network. The method we propose identifies the range of slopes for good projection lines for each pair of points and selects the slope which induces the fewest non–existent communication links. For

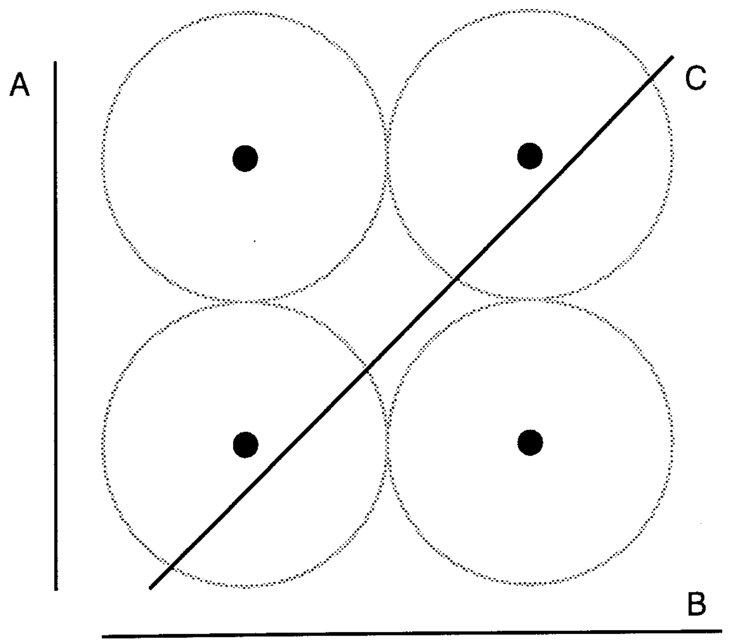

Figure 4.14: Projection lines for the network in figure 4.13

each pair of points in the plane, u and v, there is a Euclidean distance, d(u,v), given by the formula, $\sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}$. Each point also has an associated range, r_u and r_v. If d(u,v) is less than or equal to both r_u and r_v, then the choice of slope based on this pair of points will not affect the number of colors (partitions) used in coloring the linear projection of the network. The focus must then be on pairs of points for which d(u,v) is greater than either or both r_u and r_v.

Given two such points u and v, and their ranges r_u and r_v, first determine what range of slopes maintains the distance d(u,v) above either or both r_u and r_v. For example consider figure 4.15. Transceiver u is at location x_u, y_u with range r_u and point v is at x_v, y_v with range r_v. The distance between these points is $\sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}$.

The original angle between the line between the points u and v and the coordinate system x–axis is found using the following formula:

$$\vartheta_{(u,v)} = tan^{-1}\left(\frac{y_v - y_u}{|x_u - x_v|}\right)$$

The projection line slope must be close to this slope in order to keep the inequality $d(u,v) > r_u$ or $d(u,v) > r_v$ true. The range of possible slopes which do not induce a

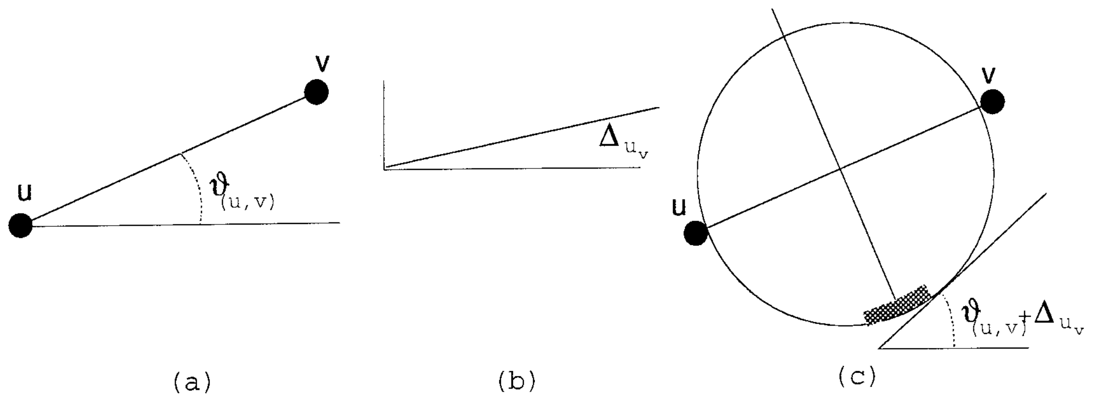

Figure 4.15: Determining good projection slopes for two points in the plane.

new communication link are found in the following way.

Define an angle of variance, Δ, between the existing slope of the line between the points and the projection line. This angle expresses limits on the difference between the original angle $\vartheta_{(u,v)}$ and the angle of the projection line. For any pair of non-communicating transceivers two angles are defined by the following equations:

$$\Delta_{u_v} = cos^{-1}\frac{r_u}{d(u,v)}$$

to maintain the distance relative to transceiver u; and

$$\Delta_{v_u} = cos^{-1}\frac{r_v}{d(u,v)}$$

for transceiver v. The slopes which do not induce new communication links are defined by the equations: $\vartheta_{(u,v)} \pm \Delta_{u_v}$ and $\vartheta_{(u,v)} \pm \Delta_{v_u}$.

Figure 4.15(c) shows an example of the range of possible slopes by highlighting an arc of a circle which corresponds to the tangent points of lines of the correct slope with respect to transceiver u. Once all the slopes have been identified, the slope of the projection line can be selected to minimize the number of induced communication links. It should be noted that it is only necessary to consider angles within one hemisphere, as angles in the opposing hemisphere will result in the same projection line.

As shown in figure 4.16, if the projection line shares the origin with the coordinate system, the location of the projection of a point is easy to determine. If α_u is the

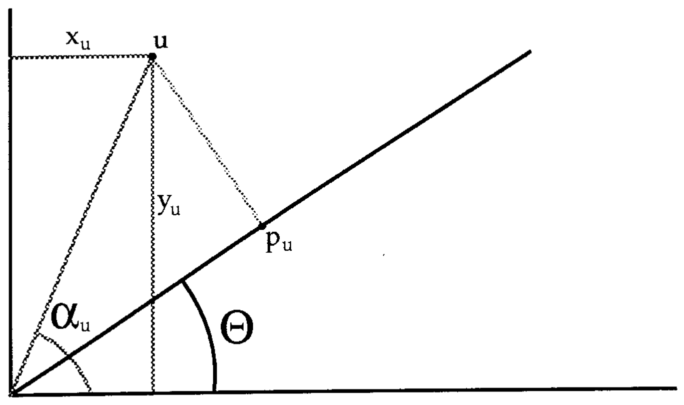

Figure 4.16: Determining the projection of a point to the projection line.

angle of line from the origin to the point, and Θ is the angle of the projection line selected, the projection line position, p_u, of the point with coordinates (x_u, y_u), is:

$$p_u = cos(|\alpha_u - \Theta|) * \sqrt{x_u^2 + y_u^2}$$

The actual location of the projection line is limited only to the existing plane of the coordinate system. While it isn't necessary for the projection line to share the origin with the coordinate system, it simplifies the computations necessary to determine the position of projection points for the transceivers.

Once the projection of points is determined, all that is necessary is to create a sorted list of left endpoints, right endpoints, and position points. With this list the points can be colored using the 1–dimensional algorithm described above to optimally color the projection (approximately coloring the planar network).

The complexity of this algorithm is dominated by the time to find the best slope for the projection line. Because the transceivers are compared pairwise in finding this slope. The complexity is $O(n^2)$.

Linear Projection Ordering

This algorithm partitions the transceivers in the area being colored by their location. As shown in figure 4.17, the transceivers are colored in the order they are projected to a line. The algorithm is analogous to the linear projection algorithm described above, except the nodes are colored in a greedy fashion based their location in the projection, and on the interference with other previously colored vertices. Only the projections of the point locations are used in ordering the nodes to be colored.

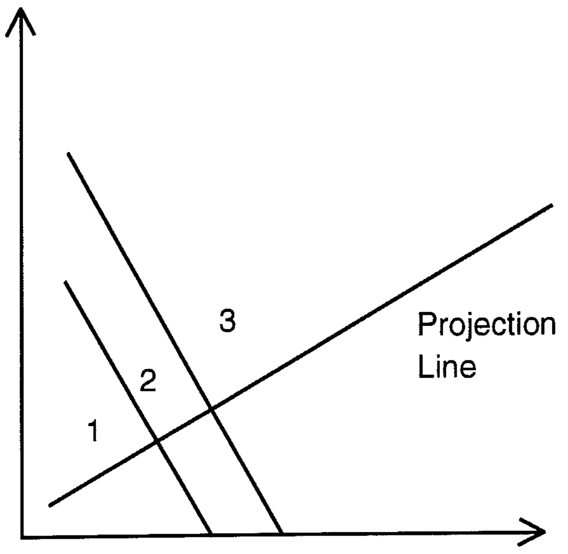

Figure 4.17: Coloring affect of linear projection ordering.

In figure 4.17, with the given projection line, the points in with locations in region 1, will be colored before those in region 2, which in turn will be colored before those in region 3. This algorithm uses the same projection line as determined in the previous section. The time complexity of this algorithm is also $O(n^2)$.

Linear Projection Order with Clustering

This algorithm is a cross between the two previous algorithms. It is similar to the first algorithm in that the points are colored in order of the left endpoints. As with the linear projection order algorithm, the nodes are colored based on the 2-dimensional location and interference.

Figure 4.18 shows why the term clustering is applied to this algorithm. By using the left endpoints to determine coloring order, it is possible to identify interfering nodes earlier and to color based on this interference. Each time a position point is reached all points whose left endpoints have been reached, but which have not already been colored will be colored.

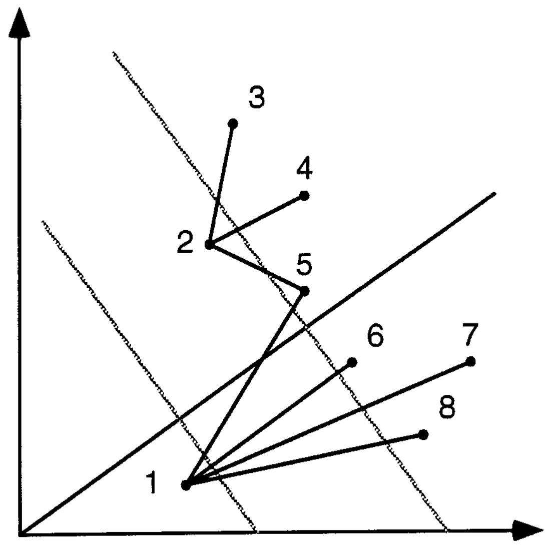

Figure 4.18: Clustering caused by using left endpoints in linear projection ordering.

As shown, nodes 1,5,6,7,and 8 will all be colored before coloring nodes 2, 3, and 4. This approach identifies secondary interference between nodes earlier, and though it is greedy, selects colors with some connection information in addition to strictly geometric information.

4.2.2 Topological Algorithms

The alternative algorithms developed return to the more traditional approach of coloring the points based on the interconnection topology. Each algorithm uses a heuristic to determine the order in which the transceivers will be colored, and uses a greedy algorithm once this order has been determined.

Greedy Matrix: A Random Algorithm

The first topological algorithm orders the transceivers according to their identification number. Since the data used in the test cases is randomly generated, this algorithm is essentially a random order greedy coloring algorithm. The name "Greedy Matrix" is derived from the use of a distance–2 adjacency matrix. For this matrix, adjacency is defined in the following way: Nodes u and v are said to be adjacent if $d(u,v) \leq r_u$, $d(v,u) \leq r_v$, or $\exists w$ such that both $d(u,w) \leq r_u$ and $d(v,w) \leq r_v$.

The time complexity of this algorithm is $O(n^3)$ due to the complexity of constructing the distance–2 adjacency matrix. If this matrix is given, the complexity reduces to $O(n^2)$.

Breadth First Ordering

Several approaches to a breadth first ordering of the transceivers are possible. In particular, the selection of the starting node, and the ordering of the adjacent nodes will affect the results of the greedy coloring. The approach selected in this work is to use the maximum degree node as the starting node. The adjacent nodes of this node are then added to the list of nodes to be colored, followed by their adjacent nodes, and so on, until all the nodes have been added to the list. Finally, any unconnected nodes are added to the list of nodes to color. Once this ordering is determined a greedy algorithm employing the distance–2 adjacency matrix described above is used to assign colors to the nodes in the order they appear in the breadth first list. This method of coloring was selected for its simplicity.

The time complexity for this algorithm is $O(n^2)$ if a distance–2 adjacency matrix, as described above, is given. If the distance–2 matrix must be constructed, the algorithm is $O(n^3)$.

Depth First Ordering

Just as with the breadth first algorithm, this is a simple algorithm which could have been implemented in several different ways. The method selected is the most straight forward. The transceiver with the maximum degree is selected as the starting transceiver. The adjacent transceiver with the maximum degree is next in the ordering. The other transceivers are placed in the list in a manner similar to the depth first traversal of a tree.

The time complexity for this algorithm is $O(n^2)$ if a distance–2 adjacency matrix, as described above, is given. If the distance–2 matrix must be constructed, the algorithm is $O(n^3)$.

Topological Algorithm #1

This algorithm is a combination of a maximum degree first and a breadth first algorithm. If the network is viewed as a topological map, the "elevation" of a transceiver in the map is directly related to its degree relative to the other transceivers in the network. This algorithm first colors the highest degree point and all points adjacent to it. The highest degree point of all remaining, uncolored points is colored next, followed by its adjacent points. This process continues until all points have been colored. This has the effect of coloring densely connected areas before coloring less densely connected areas. It is called a topological algorithm due to the analogy to a topological map in that all points adjacent to a high degree point describe something similar to an isobar on a topological map.

The time complexity for this algorithm is $O(n^2)$ if a distance–2 adjacency matrix, as described above, is given. If the distance–2 matrix must be constructed, the algorithm is $O(n^3)$.

Topological Algorithm #2

This second algorithm is similar to the first in that it implicitly creates a topological map based on transceiver interconnections. However, rather than starting with the maximum degree node, nodes are selected in a "minimum degree last" order. This is accomplished by creating a list ordered as described above except using the minimum degree nodes. This list is then used in reverse order to color the transceivers. This methodology is coincides with one of the traditional algorithms used for comparison purposes.

The time complexity of this algorithm is the same as the first topological algorithm. The overall effect of this algorithm is to change the topological representation by changing the "isobars" due to the minimum degree last ordering.

Topological Algorithm #3

As with the other topological algorithm, this algorithm is based on the combination of one of the traditional algorithms with a breadth first approach. The transceivers are ordered by distance–2 minimum degree last ordering. As with the second topological algorithm, the nodes are ordered by selecting the minimum degree transceiver first (in this case the distance–2 degree) along with its adjacent transceivers. Once the ordering is determined, the order of the list is reversed before a greedy coloring algorithm is used.

The time complexity for this algorithm is $O(n^2)$ if a distance–2 adjacency matrix, as described above, is given. If the distance–2 matrix must be constructed, the algorithm is $O(n^3)$.

Topological Algorithm #4

The final topological algorithm is included for completeness. The transceivers are selected by selecting the minimum degree transceiver first, followed by its adjacent transceivers. In a manner similar to the other topological algorithms, this algorithm creates an implicit topological map. This algorithm differs from the other three algorithms in that it colors in what could be viewed as a "bottom up" manner.

The time complexity for this algorithm is $O(n^2)$ if a distance–2 adjacency matrix, as described above, is given. If the distance–2 matrix must be constructed, the algorithm is $O(n^3)$.

Static Maximum Degree First

The static maximum degree first algorithm determines the degree of all nodes in the network and sorts them according to this degree. The term static refers to the fact that the degree of the nodes are determined before the transceivers are ordered. In all the other algorithms developed in this work, the degree of all the nodes are determined before each selection is made. Once the selection is made the degrees of the nodes in the other cases are recalculated. That is not the case here. It has the effect of reducing the multiplicative constant for the time complexity.

The time complexity for this algorithm is $O(n^2)$ if a distance–2 adjacency matrix, as described above, is given. If the distance–2 matrix must be constructed, the algorithm is $O(n^3)$.

4.3 Algorithm Comparison

This section compares the relative performance of the algorithms described in the previous section with each other, and with traditional distance–2 coloring algorithms. The algorithms are used to color networks of various densities. All the networks are defined for non–uniform transceivers. Four traditional algorithms are used in the comparison.

4.3.1 Description of Traditional Algorithms

Four traditional algorithms were modified to perform directed distance–2 coloring (source code for undirected versions of these algorithms was graciously provided by Subramanian Ramanathan [48]). The algorithms all assign the first available color to each ordered transceiver. The first of the algorithms randomly selects a transceiver to color from among all those not yet colored. This is essentially the same as the "Greedy Matrix" algorithm described above. Any performance variations in the performance of these two algorithms is due to the specific data set being colored. This algorithm is $O(n^3)$ in time complexity due to the need to check for distance–2 conflicts in assigning a color.

The second algorithm is a *Maximum Degree First* algorithm. A transceiver with the maximum degree is selected and colored, and the degree of all its adjacent transceivers are decreased before the next transceiver is selected to be colored. This dynamic determination of the maximum degree differs from the static ordering described earlier.

The next algorithm is a *Minimum Degree Last* algorithm. This algorithm produces an ordered list of nodes to color by selecting a node with the minimum degree and placing it in an ordered list. As with the *Maximum Degree First* algorithm, the degree of adjacent transceivers are changed as each transceiver selected and placed in the list. This list is then colored in reverse order of the node selection.

The final algorithm is a Distance–2 Minimum Degree Last algorithm. It is a modification of the Minimum Degree Last algorithm above. The degree of each transceiver is determined by adding the degree of each transceiver and its adjacent transceivers (therefore the name "distance–2").

All of these algorithms are $O(n^3)$ in time complexity. The complexity is dominated by the need to determine distance–2 coloring conflicts for all transceivers. As with the algorithms defined in the previous section, if a distance–2 adjacency matrix is provided the algorithms would all have a $O(n^2)$ time complexity.

4.3.2 Experimental Results

The average performance of each algorithm is compared for a series of 30 randomly generated networks. Each network is composed of transceivers located on a 200–by–200 grid. The transceivers have variable ranges from 50% to the maximum range which appears in the following tables. The estimate of the optimal number of colors is obtained by finding the maximum number of incoming communication links to any point and adding one to that number. This provides a true lower bound, but may be less than the optimal number of colors required. The results of these comparisons are shown in tables 4.1, 4.3,and 4.5.

In each table, the rows labeled "EOp" contain the estimate of the optimal coloring as described above. "LP" is the row corresponding to the linear projection algorithm developed above. This algorithm treats the linear projection as a linear network and uses an algorithm which is optimal for linear networks. "Lpo" and Lpo2" correspond to the liner projection ordering and the linear projection order with clustering algorithms described previously. "T1" through "T4" contain the results for the respective topological algorithms. The greedy matrix algorithm's results appear in the row labeled "GMt". "BF", "DF", and "SMx" represent breadth first, depth first, and static maximum degree first, respectively. The results for the four traditional algorithms are in "Rnd" for random, "Mxd" for maximum degree first, "Mnd" for minimum degree

Table 4.1: Distance–2 directed vertex coloring algorithm comparison. Averages for 100 and 200 transceiver networks in a 200–by–200 grid.

	100 transceivers					200 transceivers				
	Range					Range				
Alg	20	30	40	50	60	20	30	40	50	60
EOp	6.4	9.6	14.5	18.9	23.1	10.3	16.9	24.2	33.2	44.2
LP	23.5	32.0	39.8	47.7	53.4	43.7	58.9	75.4	89.9	106.1
Lpo	6.5	10.4	15.9	21.8	27.8	10.9	19.2	29.4	40.8	53.6
Lpo2	6.5	10.3	15.3	20.7	25.9	10.7	18.1	27.0	37.0	49.9
T1	**6.4**	10.3	15.4	20.9	26.4	10.8	18.5	27.8	39.0	52.1
T2	6.8	11.2	17.0	23.2	28.7	11.6	20.4	31.3	43.6	57.1
T3	6.6	10.5	15.5	21.4	27.1	11.3	19.3	28.4	39.3	52.5
T4	6.5	**9.9**	15.0	**20.0**	24.9	10.6	17.9	26.1	35.8	48.0
GMt	6.6	10.3	15.8	21.1	26.9	10.8	18.7	28.2	39.3	52.2
BF	**6.4**	10.0	**14.8**	**20.0**	**24.6**	**10.4**	**17.5**	25.8	35.6	47.3
DF	6.5	10.1	15.1	20.6	25.4	10.6	18.1	27.0	37.2	49.0
SMx	**6.4**	10.1	15.0	20.1	25.1	10.6	18.0	26.8	36.8	47.7
Rnd	6.5	10.6	15.8	21.3	26.9	11.0	19.1	28.3	39.5	51.8
Mxd	6.5	10.0	15.1	20.7	25.5	10.6	18.4	27.0	37.4	49.2
Mnd	7.3	10.8	16.3	20.2	24.8	11.4	18.3	**25.7**	35.3	47.2
d2M	7.5	11.0	16.4	20.4	24.9	11.9	18.8	26.7	**34.9**	**46.8**

last, and "d2M" for distance–2 minimum degree last.

As table 4.2 shows, the average deviation between the optimal coloring and the algorithm results vary as the density of the network being colored changes. For relatively sparsely connected networks, most the algorithms exhibit comparable performance. For the specific cases in table 4.2, the best general performer is the breadth first algorithm. As the density increases the performance of the minimum degree last and distance–2 minimum degree last algorithm's performance improves.

The results in table 4.1 show the performance of the algorithms on a relatively sparsely connected networks. With the exception of the *Linear Projection* algorithm, all of the algorithms produce comparable results on networks of these types. The *Linear Projection* algorithm produces colorings which use substantially more colors than the other algorithms. This result is anticipated due to the loss of information caused by projecting 2–dimensional information to a 1–dimensional space.

Table 4.2: Average percent deviation between estimate of optimal and algorithm performance (100 and 200 node networks).

	100 transceivers					200 transceivers				
	Range					Range				
Alg	20	30	40	50	60	20	30	40	50	60
LP	274.2	236.3	178.5	154.1	133.7	332.7	251.7	213.7	171.7	141.2
Lpo	1.8	7.7	10.4	15.6	20.9	6.8	13.5	21.5	23.2	21.7
Lpo2	1.7	6.8	5.6	9.7	12.5	4.8	7.1	11.4	11.6	13.0
T1	1.3	6.7	6.5	10.4	14.9	5.5	9.6	15.2	17.6	18.3
T2	6.0	17.0	17.8	22.8	25.0	13.8	21.3	29.6	31.7	29.4
T3	4.3	9.0	7.7	13.4	17.9	10.1	14.6	17.3	18.8	18.9
T4	2.2	**3.2**	3.8	6.1	8.0	3.7	6.2	7.9	8.0	8.8
GMt	4.4	7.2	9.6	11.8	17.2	5.8	11.2	16.6	18.7	18.3
BF	**0.7**	3.4	**2.2**	**5.8**	**6.9**	**1.5**	**3.6**	6.8	7.4	7.1
DF	1.9	4.8	4.3	9.0	10.5	3.1	7.3	11.8	12.3	11.1
SMx	**0.7**	4.4	3.9	6.2	9.1	2.9	6.7	11.0	11.0	8.2
Rnd	2.6	10.8	9.4	12.5	16.9	7.5	13.0	16.9	19.4	17.5
Mxd	1.9	3.9	4.7	9.4	10.6	3.0	9.1	11.5	12.9	11.5
Mnd	20.8	14.5	13.3	7.1	7.5	11.9	8.4	**6.4**	6.6	6.9
d2M	25.0	16.2	14.4	7.8	8.3	18.7	11.7	10.7	**5.3**	**6.0**

Tables 4.3 to 4.6 demonstrate the relative performance of the algorithms as the network density increases. The results indicate that certain algorithms are comparable in performance.

The *Breadth First*, *Minimum Degree Last*, and *Distance-2 Minimum Degree Last* algorithms, on average, produce results closer to optimal than the other algorithms.

The next group of algorithms contains the *Static Maximum Degree First*, *Topological Algorithm #4*, *Depth First*, *Linear Projection Ordering with Clustering*, and *Maximum Degree First* algorithms. For the most part they exhibit performance which decreases in that order, but which is comparable for all network sizes.

All the remaining algorithms except *Linear Projection*, fall loosely into a third group. They all exhibit coloring performance comparable to a random coloring algorithm, though the algorithm *Topological Algorithm #2* is consistently worse than the random algorithm, indicating an approach which is counter productive.

Finally, *Linear Projection* is in a class by itself. The number of arcs induced by

Table 4.3: Distance–2 directed vertex coloring algorithm comparison. Averages for 300 and 400 transceiver networks in a 200–by–200 grid.

	300 transceivers					400 transceivers				
	Range					Range				
Alg	20	30	40	50	60	20	30	40	50	60
EOp	13.3	22.3	34.2	46.7	63.2	16.6	29.0	43.2	61.9	82.1
LP	62.9	87.4	110.3	133.0	157.8	83.1	115.0	146.3	178.1	209.3
Lpo	14.8	27.4	42.3	60.3	81.3	18.9	35.4	56.0	80.5	105.6
Lpo2	14.4	24.7	38.7	53.7	73.2	18.1	32.6	49.0	72.4	95.6
T1	14.6	26.1	40.7	58.0	76.0	18.5	34.2	52.9	77.2	99.9
T2	15.9	28.5	45.1	64.7	88.3	20.5	38.0	58.4	85.6	113.9
T3	14.9	26.0	40.0	57.1	77.9	19.1	34.6	52.5	76.0	102.7
T4	**14.1**	24.3	37.3	52.2	70.7	18.1	32.1	47.9	70.1	91.3
GMt	14.8	26.5	40.7	57.4	76.9	18.9	34.4	53.3	76.6	100.0
BF	**14.1**	**23.6**	36.5	50.9	69.4	**17.7**	31.2	**46.9**	68.3	90.2
DF	14.2	25.0	38.8	54.6	71.3	18.2	33.2	50.8	71.9	92.9
SMx	**14.1**	24.7	38.3	53.6	70.2	17.9	33.0	50.4	71.2	91.8
Rnd	14.8	26.1	40.7	57.3	76.3	19.0	34.6	53.1	76.9	100.9
Mxd	14.2	24.7	39.0	54.5	72.5	18.0	32.9	50.2	72.7	94.9
Mnd	14.5	24.5	**36.2**	51.2	69.3	18.4	31.2	47.9	68.3	91.2
d2M	14.8	24.3	36.8	**50.8**	**68.9**	18.4	**30.5**	47.5	**67.5**	**89.8**

projecting the points onto a line have a major impact on the performance of the algorithm. This affect decreases as the density of the network increases. In a highly connected network, more transceivers have communication links, and therefore fewer links are introduced by projecting the transceiver positions to a line.

As the density of the network increases, the disparity between coloring algorithms becomes more apparent. Clearly, if a geometrically based algorithm is to be developed to take advantage of the peculiarities of planar–point graphs, it must maintain some of the information lost by the *Linear Projection* algorithm. The relative performance of the *Breadth First* algorithm would seem to suggest that an ordering algorithm which takes advantage of network connection topology might perform better. From the standpoint of distributed scheduling, this result suggests that any breadth first type algorithm can produce a reasonable schedule. For connected networks this implies that any transceiver may be assigned the task of producing a schedule, and the

Table 4.4: Average percent deviation between estimate of optimal and algorithm performance (300 and 400 node networks).

	300 transceivers					400 transceivers				
	Range					Range				
Alg	20	30	40	50	60	20	30	40	50	60
LP	378.3	294.2	223.8	185.3	150.6	405.8	299.1	240.5	188.7	155.5
Lpo	12.0	23.1	23.9	29.1	28.8	14.4	22.9	29.9	30.2	28.8
Lpo2	8.3	11.1	13.3	14.9	15.9	9.4	12.8	13.5	17.1	16.6
T1	10.2	17.4	19.2	24.1	20.5	11.6	18.5	22.8	25.0	21.9
T2	20.3	27.9	32.1	38.6	39.8	24.1	31.5	35.5	38.6	38.9
T3	12.3	16.6	17.0	22.1	23.5	15.3	19.9	22.0	23.1	25.3
T4	6.6	9.1	9.2	11.8	11.9	9.1	10.9	11.0	13.4	11.2
GMt	11.5	19.0	19.1	23.0	21.8	14.0	19.1	23.8	24.1	21.9
BF	**6.1**	**5.8**	**6.6**	9.0	9.8	**6.5**	7.8	**8.8**	10.5	10.0
DF	6.8	12.5	13.6	17.1	13.0	10.2	15.2	17.9	16.4	13.3
SMx	6.4	10.8	12.1	14.8	11.2	7.9	14.5	17.1	15.2	11.9
Rnd	11.7	17.1	19.3	22.7	20.8	14.8	19.8	23.3	24.5	23.1
Mxd	6.8	10.9	14.1	16.6	14.8	9.0	14.0	16.6	17.7	15.8
Mnd	9.5	10.4	6.0	9.8	9.8	11.6	7.7	11.0	10.6	11.2
d2M	11.6	9.3	7.8	**8.8**	**9.2**	11.4	**5.7**	10.1	**9.3**	**9.5**

resulting schedule will be comparable to a schedule produced by any other transceiver.

One final issue is of interest with respect to the linear projection algorithm. When projecting to a line, the number of additional communication links added using the "best" projection line is a concern. As previously stated, the density of the network will affect this to a large extent, and the density is a function of the number of nodes, the area occupied, and the ranges of the nodes in the network. The actual number of additional links was calculated for networks with the same parameters used in the test cases above. For the sparsest network parameters above, 100 transceivers on a 200–by–200 grid, with ranges from 10–20 units, there were on average approximately 7 times more links in the linear projection than in the 2–dimensional network. For the most dense networks, there were roughly 3 times more links in the projection than in the original network.

A comparison of the time complexity of the algorithms reveals that the form of the input determines the complexity of these algorithms. For example, if a linear

Table 4.5: Distance–2 directed vertex coloring algorithm comparison. Averages for 500 and 600 transceiver networks in a 200–by–200 grid.

	500 transceivers					600 transceivers				
	Range					Range				
Alg	20	30	40	50	60	20	30	40	50	60
EOp	19.5	34.7	52.5	74.6	99.9	22.6	39.8	61.3	86.8	121.1
LP	100.5	143.4	181.8	218.7	258.1	119.2	168.1	216.9	265.1	312.4
Lpo	23.0	44.2	69.3	100.1	130.7	26.9	51.6	83.5	118.5	157.4
Lpo2	21.4	39.7	61.0	87.3	118.0	24.9	46.0	71.8	103.1	142.4
T1	22.3	41.4	66.7	96.6	124.2	25.8	49.0	79.0	113.1	150.9
T2	24.2	47.4	73.9	104.8	140.4	29.0	56.2	88.9	127.8	167.9
T3	22.3	41.7	65.4	93.4	126.1	25.8	49.1	76.9	110.2	153.4
T4	21.1	38.7	59.5	84.0	112.6	24.9	45.1	68.8	98.6	136.5
GMt	22.4	42.0	66.0	93.5	123.1	26.1	49.0	78.1	110.2	148.4
BF	**20.9**	**37.1**	58.0	**82.7**	110.9	24.3	43.8	**67.9**	**97.0**	134.5
DF	21.7	39.8	63.5	88.5	114.6	25.2	47.5	75.2	103.1	137.3
SMx	**20.9**	39.4	63.0	88.5	112.6	24.9	47.1	75.3	102.8	136.2
Rnd	22.3	41.7	66.2	93.9	123.2	26.7	49.4	77.9	110.6	148.4
Mxd	21.5	39.8	62.8	88.8	116.7	25.1	46.7	74.3	105.7	140.7
Mnd	21.7	37.3	58.4	83.2	111.1	**23.7**	43.6	68.6	97.9	134.7
d2M	21.9	37.4	**57.8**	82.8	**110.1**	24.1	**43.0**	68.0	97.1	**133.2**

projection is provided, *Linear Projection* is $O(n \log n)$, but determination of the *best* projection line is $O(n^2)$. *Linear Projection Order* and *Linear Projection Order with Clustering* are both $O(n^2)$ algorithms if given a distance–2 adjacency matrix. The distance–2 adjacency matrix requires $O(n^3)$ time to compute. All algorithms developed in this work are dominated by this term. If such a matrix is provided, the developed algorithms are all $O(n^2)$. The traditional algorithms are all implemented as $O(n^3)$ algorithms.

4.4 Discussion of Results

The best performing algorithms in our tests are those that address topology by focusing on clusters of transceivers. In the case of the *Maximum Degree First*, *Minimum Degree Last*, and *Distance–2 Minimum Degree Last* this clustering is achieved

Table 4.6: Average percent deviation between estimate of optimal and algorithm performance (500 and 600 node networks).

	500 transceivers					600 transceivers				
	Range					Range				
Alg	20	30	40	50	60	20	30	40	50	60
LP	417.9	315.6	247.0	194.1	158.8	432.7	323.9	254.3	205.9	158.3
Lpo	18.3	27.5	32.1	34.3	31.0	19.9	30.0	36.3	36.7	30.0
Lpo2	10.0	14.4	16.1	17.0	18.1	10.5	15.8	17.2	18.9	17.7
T1	14.6	19.6	27.1	29.7	24.5	14.4	23.5	29.0	30.5	24.7
T2	24.5	36.8	40.9	40.6	40.7	28.8	41.3	44.9	47.4	38.7
T3	14.5	20.4	24.6	25.4	26.3	14.7	23.7	25.5	27.2	26.8
T4	8.4	11.5	13.4	12.6	12.8	10.3	13.4	12.4	13.7	12.7
GMt	15.0	21.4	25.9	25.5	23.4	16.1	23.5	27.4	27.2	22.7
BF	7.3	**7.1**	10.5	**10.9**	11.1	8.0	10.4	**10.9**	**11.9**	11.1
DF	11.2	15.2	21.2	18.8	14.8	12.0	19.7	22.8	18.9	13.4
SMx	**7.1**	13.9	20.2	18.8	12.8	10.8	18.7	23.0	18.6	12.5
Rnd	14.6	20.5	26.1	25.9	23.4	18.4	24.3	27.3	27.6	22.7
Mxd	10.5	14.8	19.8	19.1	17.0	11.4	17.6	21.3	21.9	16.3
Mnd	11.4	7.6	11.2	11.6	11.3	**5.1**	9.8	12.1	13.0	11.3
d2M	12.8	7.9	**10.3**	11.1	**10.3**	7.0	**8.3**	11.1	12.1	**10.1**

by coloring highly connected nodes first. For the *Breadth First* algorithm, clustering is the result of considering a transceiver in conjunction with its neighbors.

This clustering of transceivers is most apparent in the performance of the *Linear Projection Ordering with Clustering.* This algorithm consistently requires fewer colors than the traditional *Maximum Degree First* algorithm.

There is no single approximation algorithm which requires fewer colors in all cases. This is due to two factors. First, the use of a greedy algorithm means that once a conflict is discovered an additional color is used rather than attempting to avoid the conflict by reassigning the colors already used. The second factor is the relative simplicity of the heuristics used in the coloring algorithms. Clearly more work must be done to understand the relationship between the interconnections in a graph and the number of colors required to color it.

Chapter 5

Link Scheduling

This chapter examines link scheduling of radio networks. The first section discusses the complexity of link scheduling. The next section discusses the algorithms developed for prn–coloring as part of this work. The performance of these algorithms is then compared to the performance of established prn–coloring algorithms. The final section analyzes the results obtained in the previous sections of the chapter.

At this point it is appropriate to expand the discussion provided in chapter 2 regarding the reason why prn–coloring is a topic of discussion in radio network scheduling. Historically, broadcast scheduling was the mechanism used in scheduling transceivers in radio networks. For voice communication between adjacent transceivers, broadcast scheduling is still the primary mechanism. With the advent of multi–hop radio networks, it became possible to send messages to non–local transceivers by routing the message between transceivers lying on a path from the source of the message to the receiver.

As messages are sent through multi–hop networks, the information is decomposed into small packets of information which are assembled at the receiving transceiver. This process permits each packet to take a route which is independent of the route of the other packets in the message. In the most general case, each transceiver along the route of the packet must determine whether a received packet is intended for it or not. If the packet is intended for this transceiver it must be combined with other packets, if necessary, and the message must be extracted. If the message is intended for a separate transceiver, the destination must be identified, and the packet must be sent either to the destination, or to another intermediate transceiver which will send the message in the proper direction.

If broadcast scheduling were used in this problem, the message would need to contain additional information about the route as well as the final destination. In addition, broadcast scheduling would have the tendency to limit the number of packets in the network at any given time. To reduce the amount of overhead associated with each packet and to potentially allow more packets concurrently in the network, the transceivers are scheduled in a pairwise fashion. Any packet received during a scheduled time slot must be processed, and the receiving transceiver can use local routing information to determine which other transceiver to forward the packet to, should that be necessary.

Figure 5.1 shows how this can allow more packets to be simultaneously routed through the network. In figure 5.1(a), two adjacent transceivers can receive packets during the same slot in the schedule. Likewise, though they are adjacent, figure 5.1(b) shows that if there are no transceivers which can receive from two adjacent transceivers, they can send packets during the same schedule slot.

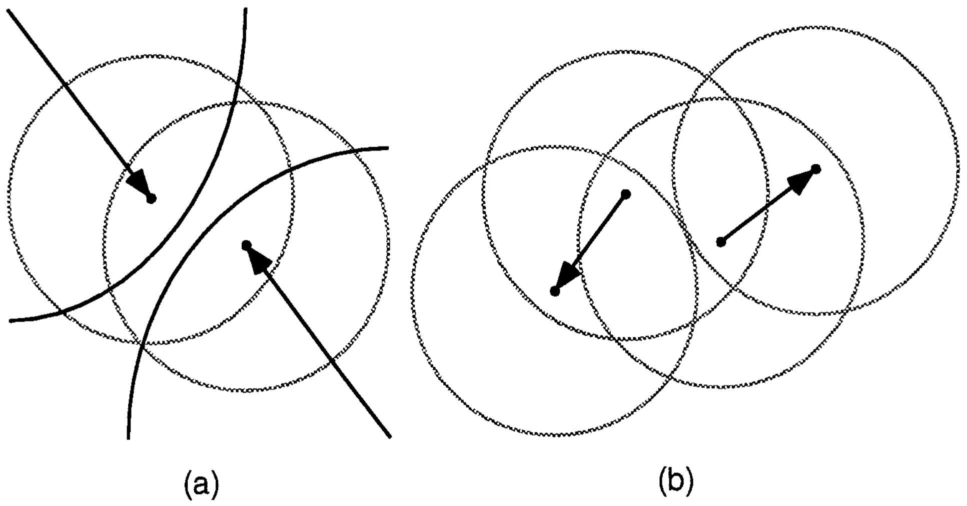

Figure 5.1: Non–conflicting schedule slots in packet radio network link scheduling.

As discussed in chapter 2, the interference definition will depend on the capabilities of the network being scheduled. However, for the simplest transceivers using TDMA, the definition of interference will be consistent with the one previously defined. Figure 5.2 shows the conditions under which two links conflict.

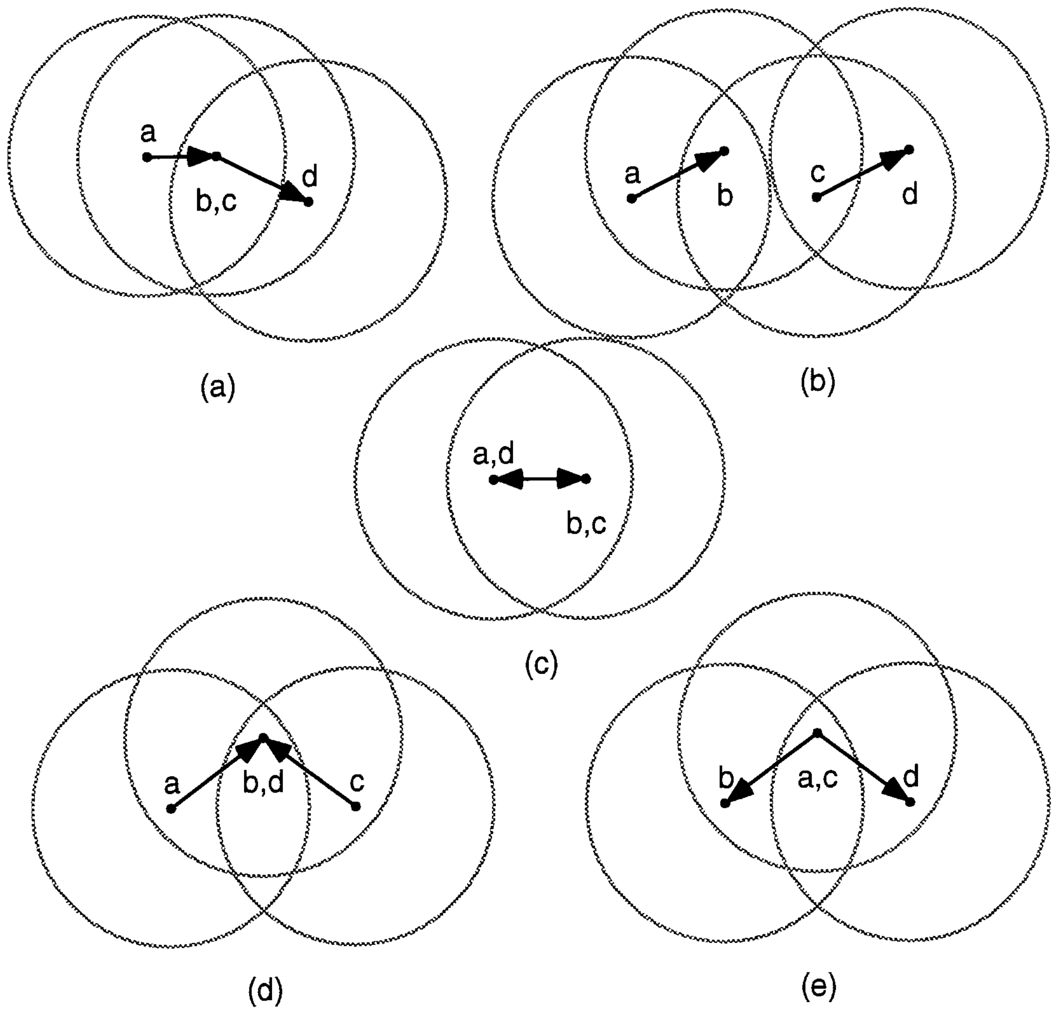

Figure 5.2: Conditions where two links (a,b) and (c,d) conflict.

The formal definition of prn–coloring is given on page 15, and is restated here for convenience.

> Given a graph G = (V,A), a *prn–coloring* of the graph is a mapping $\{\Phi : (a,b) \in A \rightarrow \mathcal{N} | \phi(a,b) = \phi(c,d) \Leftrightarrow a \neq b \neq c \neq d \ and \ (a,d),(c,b) \notin A\}$.

5.1 Complexity of PRN–Coloring Point Graphs

Edge coloring of a graph consists of coloring the edges of the graph such that no two adjacent edges have the same color (here adjacency means the edges are incident to a common vertex). By Vizing's theorem, the number of colors needed to edge color graph is either ρ or $\rho + 1$, where ρ is the maximum vertex degree. Holyer showed that determining whether a graph is ρ edge colorable is NP–complete [32].

A closely related problem in radio networks was investigated by Arikan [1]. This work shows that finding a minimum schedule satisfying link traffic demands is solvable using integer linear programming and is also NP–complete. As Ramanathan points out, the problem of prn–scheduling (as described in chapter 2) is a special case of the problem discussed by Arikan [48]. As such the NP–completeness of prn–scheduling is not necessarily a result of Arikan's work.

For complementary graphs, [21] showed k–prn–coloring can be solved in polynomial time for $k \leq 5$. They also show the problem is NP–complete for all even k ≥ 6.

Ramanathan references a proof of the NP–completeness of k–prn–coloring arbitrary directed graphs for $k \geq 3$ [48]. While this proof is not provided, he shows that for $k < 3$, prn–coloring of arbitrary graphs is solvable in polynomial time. In addition, [48, 49] contain a proof of the NP–completeness of 7–prn–colorability of planar graphs.

While these results are not directly applicable to point graphs, the proof constructs used can be adapted for use in proving 7–prn–colorability of planar point graphs is NP–complete. This proof will not be provided here since it is a direct application of the methods used in the previous proof.

The complexity of linear point graph prn–colorabililty is an open problem. The complexity of planar graph prn–coloring and the NP–completeness of edge coloring [32] give hints to the complexity of this problem. Ramanathan shows that trees and ring networks can be optimally colored in polynomial time, but it can easily be shown that linear point graphs are neither trees nor rings. The conjecture that linear point graph prn–coloring is NP–complete is stated here without further comment.

Approximate prn–coloring of linear point graphs can be accomplished in polynomial time. The algorithm described below produces an approximate prn–coloring for linear point graphs. In this algorithm, a list of available colors is maintained for each transceiver. This list is logically kept as four lists. When a color removed from the list of colors available for a point, it is identified as being used as a color for an incoming arc, an outgoing arc, an incoming secondary interference color, or an outgoing secondary interference color. Figure 5.3 shows the four conditions where a color may not be used as the color for an arc (k,i).

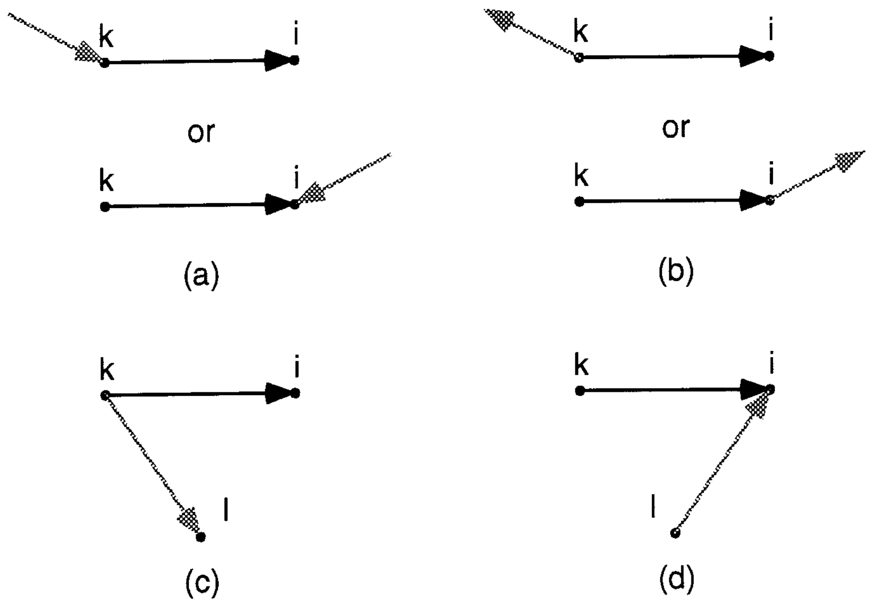

Figure 5.3: The four local conditions where a color may be unavailable for use in coloring an arc (k,i).

These conditions must be checked in the algorithm. Names are given to these conditions in the algorithm. IN corresponds to the condition shown in figure 5.3(a). OUT is used to represent figure 5.3(b). SOUT is the name for the condition in figure 5.3(c), and SIN is used for figure 5.3(d). For notational purposes, NOTIN[i,j] is true if j is not used as an incoming arc color. The name NOTIN[i,j] means "for node i, j is not an incoming arc color." Similar notation is used for the other three conditions.

```
for i = 1 to N do:
    for j = 1 to $N^2$ do:
        NOTIN[i,j] $\leftarrow$ true
        NOTOUT[i,j] $\leftarrow$ true
        NOTSIN[i,j] $\leftarrow$ true
        NOTSOUT[i,j] $\leftarrow$ true
for i = 1 to N do:
    for k = 1 to N do:
        if $(k, i) \in A$
            j $\leftarrow$ 1
            while $(k, i)$ not assigned a color
                if NOTIN[k,j] $\wedge$ NOTIN[i,j] $\wedge$
                        NOTOUT[k,j] $\wedge$ NOTOUT[i,j] $\wedge$
                        NOTSOUT[k,j] $\wedge$ NOTSIN[i,j]
                    ASGN[k,i] $\leftarrow$ j
                    NOTIN[i,j] $\leftarrow$ false
                    NOTOUT[k,j] $\leftarrow$ false
                    for l = 1 to N do:
                        if $(k, l) \in A$ NOTSIN[l,j] $\leftarrow$ false
                        if $(l, i) \in A$ NOTSOUT[l,j] $\leftarrow$ false
                j $\leftarrow$ j + 1
```

The points to be colored are assumed to be sorted in order according to their positions. This can be assumed without loss of generality, and ensures that if $i < j$, then point i is colored before point j. The time complexity of this algorithm is $O(n^4)$. This worst case complexity occurs in the degenerate case of a fully connected graph, where each arc must have a unique color. Since n^2 arcs must be colored, the "while" loop is executed $n^2/2$ times on average, meaning the condition test is executed $O(n^4)$ times. Note the inner most for loop is executed only once for each arc, resulting in $O(n^3)$ executions of the for loop body.

Expressed in terms of the number of arcs in the graph, the maximum node in-degree, ρ_{IN}, and the number of nodes, n, the time complexity is $O(|A| * \rho_{IN} * n)$.

It is easily verified that this algorithm produces a conflict free prn–coloring. For each arc, the color, j, assigned to that arc will not interfere with any previously colored arc. The four logical color lists will prevent the edge (k,i) from conflicting with any colored arc meeting the conditions shown in figure 5.3.

Colors are assigned to arcs only when the endpoint of the arc is encountered. For every point i, incoming arc (k,i) is assigned the minimum color which is not used by an interfering arc as depicted in figure 5.3.

For the first point, all incoming arcs must have different colors. For $i = 1$, the algorithm colors the incoming arcs using a number of colors equal to the in–degree of point 1. This is shown in figure 5.4(a) and (c).

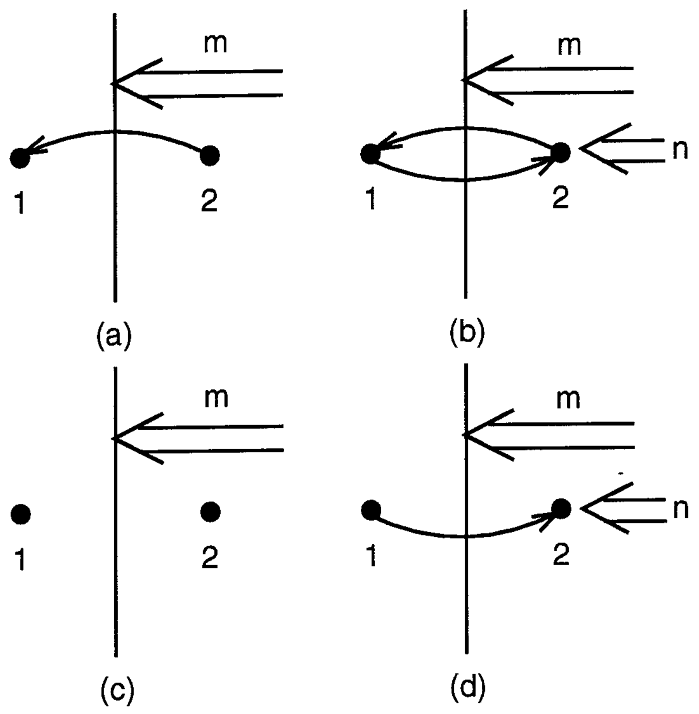

Figure 5.4: Edge coloring restrictions in coloring nodes 1 and 2. (a) all m links to node 1 must have different colors from each other. (b) as node 2's links are colored, all n links must be different from the m links to earlier nodes. This is true whether or not node 2 has a link to node 1, as shown in parts (c) and (d).

For $i = 2$, all links must have a different color from incoming arcs to point 1. This is shown in figure 5.4(b) and (d). Since node 2 is in range of the m links to point 1 it must be able to differentiate between its incoming links and those to point 1. Therefore none of the colors assigned to the n incoming links to point 2 can have the same color as any of the m links to point 1. Similarly, any links between points 1 and 2 cannot be the same as any of the $m + n$ colors used to color links (k,1) or (k,2) for

$k > 2$.

As figure 5.5 shows the same conditions hold true for all points in the linear point graph. When coloring any point, $i + 1$, a total of $m + n + p + q$ colors must be used.

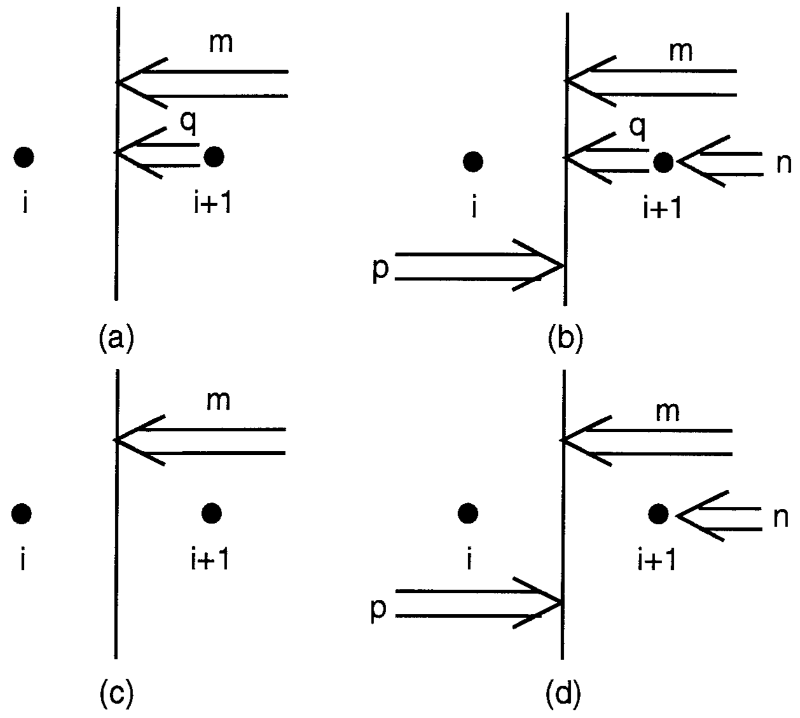

Figure 5.5: Edge coloring restrictions for coloring node $i + 1$ in a linear point graph.

This algorithm can be shown to be non-optimal. The restriction to a single dimension makes a proof method similar to a proof for planar point graphs unlikely.

Prn–coloring of D–dimension point graphs for $D > 2$ is a direct result of the NP–completeness of planar point graph 7–prn–colorability. The k–prn–colorability of all point graphs is an open problem for $3 \leq k < 7$.

5.2 Algorithm Development

Ramanathan has shown that it unlikely there exists a good approximation algorithm for prn–coloring a graph (where "good" refers to a performance bound of $O(1)$, a constant time optimal for the number of colors used) [48]. In practice, the performance of approximation algorithms for prn–coloring of planar point graphs is a concern. In this section some prn–coloring algorithms are developed to approximate the optimal solution to prn–coloring. As in chapter 4, to general approaches are taken in algorithm development. A geometric algorithm is developed to exploit the simplicity of linear point graph network coloring. Topological algorithms are also developed which rely on the network interconnections rather than the geometric locations of the points in the network.

5.2.1 Geometric Algorithm

The geometric algorithm developed for prn–coloring is based on the greedy algorithm described previously for linear point graph prn–coloring. The points in the 2–dimensional space are projected onto a linear network and the order determined by the projection is used in coloring the network. This will usually not result in an optimal solution. Just as with a linear point graph, the colors are assigned greedily to the incoming arcs for each point. However, two links which could be assigned the same color, may be assigned different colors as the associated points are independently colored in the order of the projection. These two arcs may not be independent, and could interfere with a third arc when it is colored because of secondary interference.

The time complexity for this algorithm is dominated by the time to maintain the four logical coloring lists, and is $O(n^4)$

5.2.2 Topological Algorithms

Three topological algorithms are developed for this work. Based on the topological results of distance–2 coloring, two breadth first and one depth first algorithm were developed. These were selected for their simplicity, and the connected nature of the arcs being colored. As in the linear point graph algorithm described previously, all uncolored incoming arcs are colored as each vertex is visited in the topological order. The polynomial time algorithm for coloring linear networks is used in assigning prn–colors once the order of of the vertices is determined.

The breadth first algorithm starts by coloring all the incoming edges to the maximum degree node in the network. After the maximum degree node has all of its incoming edges colored, the adjacent nodes are visited in random order. The measure of the node degree is based on both incoming and outgoing edges. The time complexity of the algorithm is dominated by the complexity of coloring once the ordering is determined. The linear network algorithm described previously is used to assign colors, and this algorithm is $O(n^4)$.

The next algorithm developed is a depth first algorithm. In this algorithm, the maximum degree node is selected as the first node to be visited. This node is then removed from degree computations for the remaining nodes, and the maximum degree node from among the adjacent nodes is selected to be visited next. This process continues, with all unconnected node being added as the last nodes to be visited. Once again all incoming arcs are colored as each node is visited. The time complexity of this algorithm is also dominated by the linear network coloring algorithm, and is $O(n^4)$.

The final algorithm is a modification of the breadth first algorithm which starts with the maximum "in" degree node rather than the maximum degree node. This algorithm was developed to determine the importance of the starting node in determining the order and subsequent coloring. The linear network coloring algorithm dominates the $O(n^4)$ time complexity of this algorithm.

5.3 Algorithm Comparison

Three existing coloring algorithms were used in comparing experimental results in this work. These algorithms were the outgrowth of the work by Ramanathan [48], and were graciously provided as test cases by him. All of the algorithms were executed on the same graphs. These test cases were averaged over 20 cases for each combination of number of node and maximum node ranges.

The first algorithm provided is a random prn–coloring algorithm. The nodes are selected randomly from among the population of nodes having uncolored arcs. All arcs incident to the node being visited are colored.

The second algorithm orders the nodes to be visited by finding the maximum cliques and coloring the arcs incident to nodes in the clique first. At each step, the maximum clique is found and uncolored arcs are assigned colors greedily.

The final algorithm implements the arboreal algorithm described in [48]. The graph is decomposed into in–oriented and out–oriented forests, and an optimal tree prn–coloring algorithm is used to color the decompositions. For further information on this algorithm the reader is referred to [48]. This algorithm has a performance bound of $O(\Theta\rho)$, where Θ is defined as the thickness of the graph and ρ is defined as the maximum vertex degree of the graph. Time complexity for this algorithm is $O(n^3 \log n)$ (given more information about the graph the time complexity can be expressed in terms of the number of edges and nodes in the graph, as $O(|E|*|V|\log|V|)$).

The experimental results are summarized in table 5.1.

Table 5.1: A comparison of prn–coloring algorithms.

Link Scheduling Results									
			Algorithms						
N	Rng	EOpt	Pfoa	Pmcf	Prbl	LPOr	BF	DFOr	MdBF
200	20	22.6	58.1	56.3	56.7	57.5	56.2	**55.7**	56.1
	30	34.2	144.6	141.1	139.6	145.8	139.6	**136.4**	137.4
	40	52.5	297.8	296.1	294.5	308.8	288.4	**285.1**	285.2
	50	99.4	528.8	524.7	516.9	547.7	508.6	**502.7**	504.3
	60	96.2	820.2	817.0	807.2	862.3	789.2	784.8	**772.6**
300	20	29.1	94.3	95.0	93.9	96.5	93.5	**92.2**	92.3
	30	49.5	261.9	259.1	257.9	268.6	253.7	**249.7**	251.0
	40	85.2	551.7	545.5	539.6	577.4	531.7	**521.6**	526.8
	50	111.2	1022.3	1004.4	998.9	1088.7	978.0	967.8	**963.3**
	60	150.4	1710.6	1722.4	1691.3	1835.7	**1622.1**	1647.7	1624.6
400	20	36.2	140.6	136.9	138.9	142.4	137.2	**134.6**	135.9
	30	62.5	390.9	381.5	382.1	403.9	380.5	**369.2**	374.3
	40	111.2	849.3	839.1	825.1	894.1	821.0	**794.8**	795.0
	50	144.4	1662.3	1650.2	1624.8	1764.3	1591.2	1581.6	**1570.6**
	60	193.1	2754.4	2757.1	2691.8	2996.8	2670.5	2663.1	**2622.7**
500	20	45.2	195.4	194.1	193.1	201.1	190.6	**187.2**	188.1
	30	74.2	531.2	530.2	531.0	547.9	518.7	**509.4**	512.5
	40	146.0	1169.8	1158.3	1142.0	1210.2	1112.4	1100.4	**1084.9**
	50	175.7	2390.1	2407.6	2350.1	2588.6	2323.8	**2298.0**	2299.0
	60	238.6	3997.0	4078.1	3906.6	4357.0	**3746.7**	4076.0	3777.7

The column labels correspond to the algorithm used. "Eopt" contains a very gross estimate of the optimal number of colors needed to prn–color the network. "Pfoa" is the random algorithm which assigns the first available color to an arc. "Pmcf" contains the results for the maximum clique first algorithm. The last of the traditional algorithms is represented by column "Prb1" which contains the results of the arboreal algorithm developed by Ramanathan. "LPOr" contains the results of the linear projection ordering of the vertices. "BF" and "MdBF" contain the results of the breadth first algorithms, with "MdBF" representing the maximum in–degree algorithm. Finally, "DFOr" contains the results of the depth first ordering algorithm.

5.4 Discussion of Results

An examination of the results contained in table 5.1 shows that the depth first and breadth first algorithms consistently outperform the more complex maximum clique, arboreal, and linear projection ordering algorithms. One reason for this is that these approaches carry dependency and conflict information each time the incoming links to a node are colored. By following graph links in the coloring algorithm a larger number of conflicting colors is discovered at each step of the algorithm. This is particularly true of early identification of secondary conflicts.

The results indicate that linear projection ordering is actually worse than a random order coloring of the nodes. This is due to the nature of the greedy coloring algorithm used as the basis of this method. As a set of multi–dimensional points are ordered along a single projection line the single dimension information indicates a measure of "nearness" between adjacent points. Thus adjacency, while not indicative of primary interference, makes secondary interference more likely to occur between the points. The independent coloring of links which cause secondary interference increases the inefficiency of the greedy algorithm. Links which could have the same color, but interfere in the coloring of other links, are more likely to have different colors since the interference is not considered when the links are independently colored. If some dependence or conflict information was available, as in the depth first and breadth first algorithms, they would be more likely to have the same color. Linear projection ordering increases the occurrence of this type of situation by collapsing multiple dimensions into a single dimension.

The depth first and breadth first algorithms outperform the arboreal algorithm for the same reason. By decomposing the graphs into multiple in and out oriented forests, the dependency and conflict information imposed by the graph links is lost when coloring the decompositions independently. Just as with the linear projection ordering algorithm, this delay reduces the efficiency of the greedy algorithm used.

Single algorithm results are somewhat unpredictable based on the test cases used. For example, while producing the best average results on most test cases, the depth first algorithm had an average number of colors nearly 9% worse than breadth first for the densest (most highly connected) graph. A possible reason for this is that as the density of the graph increases the number of cliques in the graph increases, making a breadth first algorithm less likely to independently color links which exhibit a secondary conflict as described above.

In conclusion, a depth first or breadth first ordering of the points in the graph is empirically better than a more complex approach in prn–coloring of multi–dimension point graphs.

Chapter 6

Distributed Scheduling

This chapter discusses the development of distributed scheduling algorithms. As shown in previous chapters, the problems of broadcast and link scheduling of point graphs are both NP–complete. This is the impetus for using approximation algorithms in determining schedules. As with other computationally intensive problems, a desirable approach is to increase the processor power applied to the problem. This is one of the chief reasons for examining distributed scheduling algorithms. By distributing the work of determining a schedule, more computation can be done per unit time. This added computation has a cost in terms of the need to coordinate the activities of the processors working on the problem.

A second reason for distributing the scheduling process is that the network itself is spatially distributed. It is this distribution which makes it possible to share the communications medium, and the sharing which makes scheduling necessary. If the schedule is determined in a centralized location it becomes necessary to distribute the schedule. However, if the schedule can be determined locally, no such distribution is necessary.

Another reason for distributing the computations required to perform scheduling arises in mobile radio networks. As the transceivers within the network move, the interconnections between them are established or lost due to changes in their relative positions. Two transceivers may lose and establish their communications links frequently over the course of time. If scheduling were to be controlled by a single, centralized algorithm, the information about transceiver communication links would need to be communicated to the central location to determine the new schedule. Another reason to distribute the scheduling algorithm is to allow local transceivers to schedule themselves based on the interconnections they have. This makes it unnecessary to send such information to a central processor.

A final reason to distribute the scheduling process is to create a more reliable system. In a system with a centralized scheduler a transceiver failure at the scheduler causes a network failure. In addition, any transceiver failures during schedule distribution can leave portions of the network unscheduled.

For the rest of this chapter the terms, transceiver, node, and processor will be used interchangeably in reference to the network. "Transceiver" will be used primarily in reference to radio communications. "Processor" will be used when referring to the computational agents. "Node" will be used to refer to the combination of processor and transceiver. All network nodes are assumed to have identical processing capabilities and to be involved in distributed computations.

The next section discusses some of the problems associated with distributing the scheduling process among the nodes to be scheduled. The second section defines a basic set of characteristics for transceivers in the network, and presents an algorithm for distributed scheduling for a given set of characteristics.

6.1 Problems in Distributed Scheduling

Several authors have investigated the process of scheduling in a distributed environment [3, 54, 51, 13, 20, 34, 50]. In this section the issues and methods associated with distributed scheduling are discussed within the context of their work.

One issue to be raised before discussing "problems" is *the* problem. What specifically is being scheduled. As mentioned in chapter 2, the access method and the transceiver capabilities determine the characteristics of the schedule to be created. Of particular interest is the impact of the access method choice on the definition of interference.

In spread–spectrum networks, transceivers are assigned *orthogonal* codes in a transmitter based, receiver based, pairwise, or hybrid scheme [34]. The choice of scheme used determines what types of interference can be tolerated, and therefore affects the assignment of codes. Transmitter and receiver based schemes assign codes based on the transmitter or receiver sending or receiving a message. Two transceivers which have no common transceiver within their range (and which are not within each other's range) may have the same code in transmitter based schemes. For receiver based code assignments, all receivers within the range of a specific transceiver must have different codes. Both of these are equivalent to the distance–2 coloring problem described in chapter 3.

A pairwise code assignment is equivalent to the edge coloring problem. In this case no two adjacent arcs can have the same code. This problem differs from the link prn–coloring problem from chapters 2 and 5 in that the secondary interference shown in figure 1.4 is allowed. This is due to the orthogonal nature of the codes assigned [34].

The type of schedule to be created will determine what information is needed at each network node in order to determine a schedule. Adjacent transceiver information is needed regardless of the type of schedule to be created. For transceiver or receiver code assignment in spread–spectrum communication, or broadcast scheduling using TDMA or FDMA, distance–2 neighbor information is needed.

6.1.1 Basic Assumptions

The algorithms and protocols appearing in the literature define a set of assumptions about the nodes in the network. One the biggest issues in network scheduling is synchronization. Whether a centralized algorithm or a distributed algorithm is used, each node in the network must maintain an accurate global time. This assumption is common to all network protocols and algorithms discussed in the literature, and is usually not explicitly stated. The timing information can either be transmitted on a separate signal or may be transmitted as part of the message [14]. This particular issue is beyond the scope of the work in this dissertation, and the interested reader is referred to [14] or other texts on data transmission.

Probably the next most important assumption concerns the stability of the network topology. Network topology can change due to several factors. Many algorithms depend on a set of stationary transceivers [48, 50, 20], however, mobile transceivers are becoming an issue of greater concern in the research [3, 54, 33, 34, 51, 13]. It is reasonable to assume that node motion will have a greater impact on the stability of the network topology than any other single issue (of course this expectation is limited to those networks with mobile transceivers).

A second issue likely to affect network topology is transceiver failure. Networks should be able to exhibit "graceful degradation" in the presence of transceiver failures. It is reasonable to expect small failure rates, however the network should be able to continue operation when a node fails or when a node is repaired. The return of a repaired node and the addition of a new node to a network are similar operations and the definition of the communication protocol and scheduling algorithm will determine if they are treated differently (a likely difference between these two cases will be whether or not message traffic intended for a node is waiting for delivery when a node is repaired).

Another issue affecting the network topology is the range of the transceivers. Most previous work has depended on all communication links being bidirectional (i.e., modeled by a symmetric graph), implying a uniform range transceiver network. Given a set of uniform transceivers, it is likely that bidirectional links cannot be maintained [3].

A final issue affecting the network topology in the environment in which the network will operate. Many researchers have assumed the best regarding the environment (i.e. that the environment will not affect the topology or operation of the network). Because most radio communications rely on line of sight transmission, the geography of the surroundings will determine whether two otherwise in range transceivers have a communication link. For example, there may be a mountain or a group of buildings in the way. Weather is another environmental element which can affect the topology of the network by reducing the effective range of the transceivers.

Another common assumption about the network involves what information each transceiver has about network connections at the start of the algorithm. The amount of information can vary from no information to complete connectivity information. Complete connectivity is necessary in cases where the distributed algorithm simply replicates a centralized scheduler at each node. This degenerate case is attractive from the standpoint of the communications needed to distribute the schedule, but becomes impractical if the network nodes are mobile.

At the center of the set of assumptions is the set of capabilities of the nodes. The most common assumption is that the node can only do one thing at a time, at least with respect to transmitting or receiving a message. For the most part, the time to process a message (determine if this node is the destination of the message, if not prepare the message for forwarding towards its destination), is assumed to be zero. This assumption can be made if message processing can overlap with message transmission or reception, and is an artifact of the protocol used.

The more assumptions made about the network the more clearly a distributed algorithm can be defined. The assumptions made can have the effect of simplifying the problem. For example, the assumption of loss free communication links, as in [48], makes message acknowledgment, retransmission, or other methods of recovery unnecessary. On the other hand it is often possible to make the assumption that some issues, such as missing message packets, are handled at another level of abstraction. Therefore, they need not be considered in a scheduling algorithm or as part of the communication protocol (e.g., they may be handled as part of normal message processing). In the absence of any assumptions regarding the network few useful conclusions can be drawn. In designing a network, the complexity of organizing the communications process must be controlled to allow message traffic to be exchanged.

6.1.2 Network Control Options

As stated previously, the type of schedule being developed will affect the information required at each node. In addition, the method used to obtain information which is not present initially in a node must be considered. Several approaches have been used to initialize and control network communications.

The most common method of controlling node communication prior to the creation of a usable schedule is the use of a token [48, 50, 20, 51]. An alternative to token passing is the use of carrier detection to avoid collisions [56]. Unfortunately, carrier detection does not prevent message collision due to the hidden terminal problem (an out of range transceiver which can transmit to common receivers, secondary interference in figure 1.2), so a protocol must be instituted to detect and recover from message collisions. Hu offers another alternative called *deadlock free orientation* which allows nodes which have information about their adjacent nodes to exchange information without collisions [33, 34].

Another issue in controlling the network is how control information is exchanged. Two basic alternatives exist. First, a separate control channel can be used to exchange control information. This has the advantage that control messages will not interfere with normal message traffic in the network. However, the complexity of the nodes must increase to make it possible to communicate on the separate channel. In addition, it is necessary to schedule the communications on the control channel. In a network with a frequently changing topology, control channel scheduling may be either to expensive computationally, or it may be simplified to the point that the control channel is poorly utilized (both of which are relevant issues for any schedule).

The second method exchanging control information is to treat control messages as part of the normal message traffic within the network. The only requirement then, is to recognize when a control message is received or must be sent.

Regardless of which method is used to exchange information, each control message must contain information necessary to construct a usable schedule. Each node must be aware of the nodes it can receive messages from and send messages to, and for broadcast or prn–scheduling it must also be aware of the distance–2 neighbors.

Control messages must be able to provide neighbor information. This is especially important in networks where the transceivers are mobile or where the network must operate in the presence of failures. As a result each node should have a unique identifier. If the maximum number of nodes a network will ever contain, N, is known, $\ln_2 N$ bits will be necessary to represent any identifier. Depending on the format of the control messages, neighbor information may be expressed as list of these identifiers or as a bit vector of length N [54]. The network design may require N to be a large number. The network designer must decide if a message N bits long is necessary, or if the average degree of nodes in the network ρ, makes a $\rho \ln_2 N$ message length more practical.

The other information needed in control messages, is scheduling information. Knowing all distance–1 and distance–2 neighbors is sufficient to determine a schedule at a particular node. However, the schedule must be sent to the other nodes. The other nodes will either follow the received schedule or reconcile it with a locally determined schedule. As with neighbor information, the method used to encode the information must be determined by the network designer, and will not be discussed as part of the algorithm design in this dissertation.

One requirement for the exchange of control information is that every node in the network be capable of sending messages to at least one other network node. This assumption is not required in the presence of several other assumptions, namely: no message loss; every node knows the identity of all adjacent nodes; nodes never fail; and nodes are stationary. If messages can be lost, it must be possible to request a retransmission of the message. If nodes must discover the identities of adjacent nodes every node must be able to notify some other node of its existence. If nodes can fail, other nodes must be made aware of the failure and possible return to operation. Finally, if nodes are mobile, they must be able to notify at least one other node of the new nodes from which they can receive messages, to avoid the secondary interference shown in figure 1.2.

6.2 An Algorithm for Distributed Scheduling

This section describes the development of a distributed algorithm for determining broadcast schedules in a mobile radio network. First the basic assumptions about transceiver capabilities are explained. Then a distributed algorithm for a stationary network is defined. Finally the algorithm is modified to account for node movement, node failure, and the addition of a node to the network.

As mentioned previously, some set of assumptions about the network must be made in order to discuss an algorithm. For the algorithms in this dissertation a common factor has been the use of point graphs as the model of the system. This model will also be used here without restriction. The topological assumptions for the network must account for one direction communication links.

The assumptions about the network to be scheduled are:

- Every node has a unique id, assumed to be a positive integer.
- Every node knows its own identification, the identification of all its adjacent nodes, and the lowest identification number in the network.
- Every node knows the direction of its communication links.
- All communication links are reliable.
- The network is strongly connected.

The first assumption is reasonable. Most hardware manufacturers include some form of identification in their products. Node identification ensures that messages are sent to the correct destination, and are also necessary to properly route messages between non-adjacent nodes. The second assumption can be accomplished by means described in various works [3, 54, 34]. A method described in [3] can be used to determine the direction of the communication links between nodes. The assumption of reliable communications, as stated above, may be made based on the assumptions of the level of abstraction which handles message acknowledgment and requests for retransmission. Error detection and recovery methods are described in [7].

The final assumption regarding a strongly connected network is a reasonable expectation for radio networks. A strongly connected network is one where a path exists from each node to every other node. In the absence of such an assumption there can be nodes in the network which are unable to communicate with some subset of nodes in the network. This is not a desirable characteristic, and is the reason for the assumption.

A simple centralized greedy algorithm for determining this coloring is the following:

```
Given a graph G=(V,A)

While there is a node which has not been colored
    select an uncolored node, v
    assign color c(v) ← Non–conflicting–color(G)
```

The function Non–conflicting–color(G) returns the first color which does not conflict with any of the node colors already assigned for graph G. If the maximum node degree is ρ, the function requires $O(\rho^2)$ time to process, and the total algorithm completes in $O(|V|\rho^2)$. The algorithm uses at most $O(\rho^2)$ colors, and since this is distance–2 coloring, at least $\Omega(\rho + 1)$ colors.

In chapter 4 the breadth first algorithm exhibited an average performance which makes it a good choice for implementing in a distributed environment. The problem is how to ensure that nodes at the same distance from the first node colored do not interfere with each other when they communicate. The usual way to prevent this interference is to pass a token during the development of the schedule. For a stationary network, where the schedule is only determined at network startup, this is a satisfactory solution. In situations where the network may need to be rescheduled periodically, token passing is wasteful. The algorithms described in the rest of this chapter rely on two assumptions about the network. First, each node knows the identities of adjacent nodes. As stated above, this is a reasonable assumption at startup since several authors have indicated methods of distributing this information among the nodes. The second assumption regards the assumption of reliable, loss free communication links. The algorithm presented here depends on the ability to detect errors due to possible message collisions at the receiving node. An extension of this assumption, is that message reception is guaranteed in a finite time.

The distributed algorithm for a static network is now informally presented. Consider the network shown in figure 6.1(a). Assume node a is selected as the first node to be colored. This node may be selected as the node with the minimum identification number in the network. Node a selects a the minimum color available, 1, for itself, and assigns colors to each of its adjacent nodes. It then broadcasts this color assignment to all of its neighbors. These neighbors then have some distance–2 information, and an assigned color.

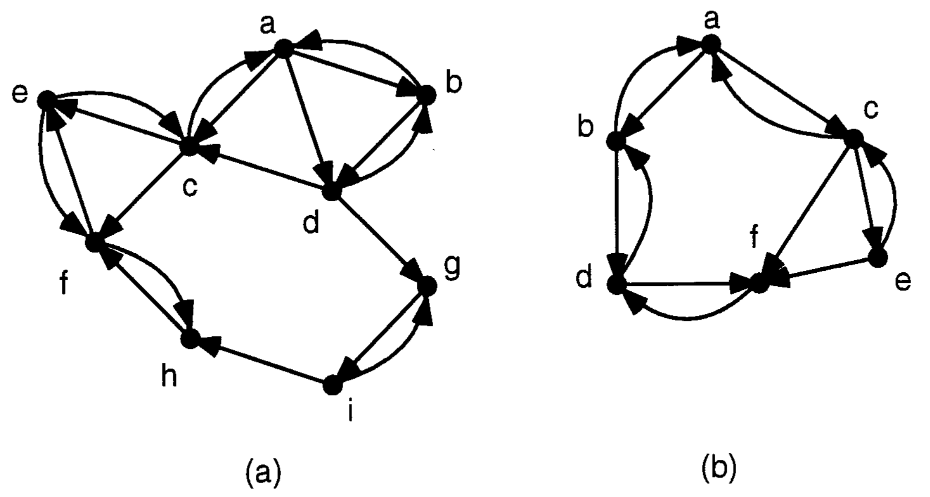

Figure 6.1: Example networks for distributed coloring algorithm

The adjacent nodes then add to the received schedule by assigning color to adjacent nodes in the following way. Wait until coloring information has been received from all adjacent nodes which are already colored and which have a lower identification color. If the received schedules assign multiple colors to this node, select the first non-conflicting color. Based on all received coloring information, assign colors to all uncolored adjacent nodes. If two nodes have been assigned conflicting colors, assign the first non–conflicting color to one of the nodes. Then broadcast the new coloring information to all neighbor nodes.

After initially processing schedule information, each node participates in the network, and may receive schedules from its neighbors. If a received schedule is different for any previously scheduled nodes, the node will update its own schedule, correcting any conflicts between the received schedule and the node's current schedule, and then broadcast the updated schedule. If there are no changes, update any link information for adjacent nodes (this accumulates distance–2 information for the node). If all adjacent nodes have finished processing, start using the schedule.

A walk through an example will show how the algorithm works. In figure 6.1(a), assume node a is the node with the minimum identification number in the network.

Node a assigns color 1 to itself, and colors 2, 3, and 4 to nodes b, c, and d. Thus after the first step, node a has created the following coloring message and sent it to b, c, and d.

a	b	c	d
1	2	3	4
–	in/out	in/out	out

The first row contains adjacent node identification, the second contains the coloring, and the third contains link direction information. Node b and c then construct their schedules based on the received schedule. Node b makes no changes to the schedule, since all adjacent nodes have been colored, but it does change the communication link information to reflect its own links, and sends this message to a, and d.

a	b	c	d
1	2	3	4
in/out	–	in/out	–

Node a will not make changes to its schedule, but records the communication information for node b. Node c does make additions to the schedule based on its neighbors, and sends the following information to a, e, and f.

a	b	c	d	e	f
1	2	3	4	2	1
in/out	–	–	in	in/out	out

When node a, receives this message, it updates its schedule to reflect the new link and coloring information and sends new copies of the schedule to b, c, d. In the meantime, node d, which waited for a schedule from node b, updates its schedule by coloring adjacent node g. It sends the following message to nodes b, c, and g.

a	b	c	d	g
1	2	3	4	3
in	in/out	out	–	out

This process will continue with nodes coloring adjacent nodes, until nodes f and i send the following messages to node h:

from node f

a	b	c	d	e	f	h
1	2	3	4	2	1	4
–	–	in	–	in/out	–	in/out

from node i

a	b	c	d	g	h	i
1	2	3	4	3	2	1
–	–	–	–	in/out	out	–

Node h must perform two actions depending on the timing of the messages. In one case, the message from i is received first. Node h will accept color 2 for itself and assign the color 3 to node f, and then send node f a message with this information. Node f will delay until a scheduling message is received from e before updating local schedule information. Node f will then resolve the coloring conflicts between nodes e and h, and between node i and itself. First it assigns itself the first non–conflicting color, in this case 4, and assigns the first non–conflicting color to either e, or h. In this particular example, the color 5 would be assigned to either e, or h.

In another case, node h receives messages from both nodes f and i before attempting to create a schedule. In this case, node f will accept the first non–conflicting color, 4, from the message received from node f. The conflict between nodes f and i will then be resolved either by assigning the color 2 to node i, or assigning 5 to node f.

A couple of important things to note about this algorithm are that message collisions are assumed to be handled at some other level of abstraction. This is important in understanding the non–deterministic order of messages received at node h in the example just discussed. Those nodes which have multiple paths from the starting node will receive multiple schedule messages. As seen in the example the message order will determine the coloring assignment in the network. However, the schedule produced will ultimately be conflict free. As a node receives a schedule message and discovers conflict, it changes the schedule and sends the new schedule as a message. This schedule traffic continues until no changes are made to the schedule, and all scheduled nodes have acknowledged the change. Message acknowledgment is necessary to ensure termination. The messages described above can include an additional row of acknowledgments. The acknowledgment for a node is set when it broadcasts its first schedule message, if the only change in the schedule message seen at an adjacent node is an acknowledgment, the schedule message should be forwarded, and the local processor will keep track of each node appearing in a schedule message, whether that node has acknowledged a schedule by sending its first message.

In the example given above, node i will receive a correct schedule and begin following it when it receives a schedule message with an acknowledgment from node h. In a network with stationary transceivers, the network will have a conflict free schedule when the schedule no longer changes. All nodes will be aware of the correct schedule when the acknowledgment for all nodes in the schedule has been received. Because the network is assumed to be strongly connected, each node will eventually have all acknowledgments. The acknowledgment portion of the message is similar to the use of a token.

An examination of figure 6.1(b) reveals the same sort of conflict problem arises for this network at node f. In both cases the multiple paths from the start node to other nodes in the network cause the conflicts. One way to avoid this is to use token passing during the development of the algorithm. The difficulty with token passing is, in a directed network where symmetry is not guaranteed, a token protocol insuring no conflicts may spend large amounts of time passing the token. The approach taken in this work is analogous to using multiple tokens by using the acknowledgment from adjacent nodes. However, processing of scheduling messages can continue while waiting for acknowledgments.

Each message is assumed to have header information including:

- source – the identification of the node sending the message
- schedule – the schedule as depicted above including identifiers, assigned colors, and direction of the communication links for the source's adjacent nodes
- acknowledgment – a flag, as described above, indicating that a node has processed a schedule message

Without loss of generality, each message can also be assumed to have a destination field in the header. The destination for schedule messages may be a special value indicating a schedule message to all adjacent nodes. Another possibility is that each message will have a "type" field, which indicates whether or not a message is a schedule message. There must be other types (at least one), but they are not of interest to this algorithm.

Each node also maintains a list of its neighbors, and the directions of the communication links. It also keeps a list of the distance–2 neighbors identified in schedule messages received from adjacent nodes. A value is also maintained locally for the acknowledgment flag of each node. Finally, each node maintains a queue of messages.

The following algorithm is executed by each node in the network:

```
if the id of this node is the lowest identifier in the network
    select colors for this node and its adjacent nodes
    set the acknowledgment flag for this node
    send the schedule to all adjacent nodes
repeat
    wait until a message is received
    if the type of the received message is "schedule"
        add message to queue
        if this node has not set its acknowledge flag
            if all nodes scheduled in the first schedule message in the queue,
                    which have lower ids and which can transmit to this node
                    have their transmitted a schedule
                for each schedule message in the queue
                    add missing schedule contents from message to local schedule
                    add distance–2 information for source of message
                    add acknowledgment flags from the message
                if this node has been assigned multiple colors
                    assign first non–conflicting color to this node
                    modify schedule to reflect new color
                for each node in schedule
                    if the node color conflicts with another node in the schedule
                        assign first non–conflicting color to that node
                        update schedule before checking the next node
                set acknowledge flag for this node
                send the schedule to all adjacent nodes
        otherwise this node has set acknowledge flag
            add missing schedule contents from message to local schedule
            if this node has been assigned a new color
                assign the new color if it does not conflict otherwise
                        assign next higher non–conflicting color
                change local schedule to reflect received schedule and changes
            for each node in schedule
            if the node color conflicts with another node in the schedule
                assign first non-conflicting color to that node
                update schedule before checking the next node
        if the local schedule or acknowledgment flags have changed
            send the schedule to all adjacent nodes
    otherwise this is not a scheduling message
        Process messages
continue repeating while the network is operating
```

Each node will send a schedule message changing its acknowledgment value exactly once. The acknowledgment values provide an end-to-end method of acknowledging the completion of the scheduling process. For stationary networks, the forwarding of scheduling messages when acknowledgment flags change allows feedback between nodes whose schedules may interfere. For example, in the walk through of figure 6.1(a), this forwarding of messages is what guarantees that node i's message was received by node h.

The following is a description of a method for handling mobile network nodes in this algorithm. There are two cases when two transceivers enter each others range. First, they may already have been distance-2 neighbors. In this case they must reconcile any new conflicts due to new distance-2 neighbors. In the second case, they may have had no previous connection in the network, and must provide the same distance-2 information to determine schedules. The main difference in these two cases will be the number of potential conflicts between the nodes. The second case is more likely to have conflicts due to the independence of the nodes prior to their movement into each other's range.

For both these cases, define a message type "request-schedule". For this work the requesting node sends a copy of it's schedule to the node it has received a new message from (i.e., one that is newly adjacent). To reduce the rate of growth of the schedule, the node with the minimum identifier is assumed to take precedence in scheduling the newly adjacent nodes. When a request-schedule type message is received from a node with a smaller identifier, entries in the local schedule are changed to avoid conflicts with the schedule received in the request-schedule type message. This precedence is important in the event two nodes both send request–schedule type messages.

The other condition which must be detectable is when two nodes are no longer within range of each other. This situation is similar to node failure. In either case, when a message is sent, some level of the network protocol will expect an acknowledgment. When the acknowledgment is not received within a delay period, the destination node can be assumed to no longer be within range, or perhaps no longer functioning. In these cases a node should remove the node from the local adjacency list (assuming of course that there was an incoming link from the node). Connectivity information for nodes in the network can be maintained by protocols discussed in [3, 54].

The mobile network algorithm described below assumes some method for identifying communication links in the network. Detection of a node moving into range can be detected by examining the header information of each received message. If the source of the message is not currently in the adjacent node list for the node receiving the message, then the source is a newly adjacent node. The schedule slot in which the message was received will indicate the color of the node.

This described algorithm should be part of normal message processing indicated in the algorithm above:

```
if the id of the message source is new and message type not schedule–request
    determine the color/schedule slot of the source node
    add source node as an adjacent node
    if this node id is less than the id of the source
        if source node conflicts with the local schedule
            assign first non–conflicting color to source node
    send local schedule in schedule–request type message to source
otherwise if message type is schedule request
    add source node as an adjacent node
    if this node id is less than the source id
        for every node conflict between local and source schedules
            assign non–conflicting color to source node and neighbors
    otherwise source id is less than this node id
        for every node conflict between local and source schedules
            assign non–conflicting color to this node and neighbors
    send new local schedule as schedule message
```

For nodes which are no longer within range, the following should be performed under conditions indicating a node is no longer within range:

```
    if node i is no longer in range, remove node i from adjacency list
```

When a node is no longer in range, schedule changes are not necessary. If the node was removed in error, it will not be discovered until a schedule change occurs. One alternative to this method is to use one of the protocols described in [3] to keep connectivity information current. The trade off is the cost of control information. If the network uses a separate control channel, control messages will not interfere with normal message traffic. The algorithms above do not make any assumption regarding the use of a control channel.

The algorithms presented here have not been analyzed with respect to efficiency. In a network with frequent connectivity changes, the algorithm will probably prove to be very inefficient. The goal of this work was to describe a distributed algorithm which can operate in a mobile environment, and which uses some of the results observed in centralized algorithms for broadcast scheduling. The topic of distributed scheduling is large, and a complete analysis is beyond the scope of this dissertation.

6.3 Conclusions

The issues surrounding distributed network scheduling were discussed in this chapter. A simple distributed algorithm for scheduling a network of stationary nodes was presented. This algorithm was evaluated for correctness, though efficiency was not addressed in its evaluation. While most algorithms described in the literature use token passing to control development of the schedule, this algorithm uses a breadth first approach in developing the schedule.

Many issues regarding distributed scheduling were discussed prior to developing the algorithm to provide a context for the algorithm. However, many of the issues raised were not discussed with respect to the developed algorithm. The issues require considerations in network protocol and transceiver design, and are outside the scope of this dissertation.

Distribution of scheduling does not change the NP–completeness of the scheduling problem. When the problem is distributed, the information needed to determine the schedule also must be distributed. In the presence of mobile network nodes, there is a continual need to exchange and update information in order to maintain a conflict free schedule. A unique feature of this algorithm is that it is designed to operate in network where node communications may not be bi–directional. Several issues which should be evaluated for this algorithm include: the amount of control message traffic; the propagation of schedules through the network; a bound on the growth of the schedule. These are left as directions for future research.

Chapter 7

Conclusions and Future Directions

This chapter presents a synopsis of the research performed for this dissertation. The first section summarizes the material presented in this work, highlighting the main contributions. The final section presents recommendations for directions to explore in future research.

7.1 Summary and Conclusions

This work addressed the problem of scheduling communication in radio networks. Previous work failed to recognize the uniqueness of the graphs used to model networks. This work showed the uniqueness of the graphs produced, the correspondence to physical networks, and the complexity of determining schedules for the network.

Chapter 1 gave a brief introduction to the problem area. Radio communications are becoming more common as the twenty–first century approaches. In particular, networks of transceivers forming packet radio networks are a continuing topic in research. The properties of radio communications make interference between transmitters a problem necessitating scheduling of the transceivers comprising radio networks. Two types of network schedule and the types of interference they experience were defined. In broadcast scheduling each transceiver transmits to all transceivers within its range. These broadcasts are scheduled so that primary and secondary broadcast interference do not occur. In link scheduling, the communications between transceivers are scheduled in a pairwise fashion. Though the definitions of interference differ, primary and secondary link interference are not allowed.

The reason for modeling of radio networks is to capture the pertinent features of the networks in order to solve the problem under consideration. In the area of network scheduling, graph models combined with graph vertex and edge coloring are used to solve the problem. Chapter 2 explores the problem of modeling in general, and the specifics of modeling radio networks.

All graph models start from the same definition of position and range. The resulting graph is identified as a particular type of graph, and this type of graph is the model used. Most research has concentrated on modeling using arbitrary graphs, though trees and planar graphs have both been proposed as network models. The selection of the type of graph used to model the network has usually been outside the interest of the researchers, resulting in the selection of arbitrary graphs. The restricted graphs are proposed as models due to the existence of polynomial time algorithms for solving the graph coloring problems used in determine network schedules. All of these graph types are shown to be inaccurate network models. The inaccuracies lead to inefficient solutions and possibly erroneous conclusions based on applying characteristics of a large problem domain a restricted domain where the characteristics do not apply.

An accurate model of radio networks, the *Point Graph*, is defined in chapter 3. While defined for N–dimensions, most networks are modeled using the 2–dimensional *planar point graph.* These graphs accurately model radio networks by representing the communication links between transceivers. The relationship between this new graph type and other graphs is shown. *Point graphs* are a proper subset of graphs and are not equivalent to trees or planar graphs. When restricted to a uniform range, point graphs are equivalent to space graphs, which are called unit disk graphs for 2–dimensional space. Proper identification of the type of graph used to model the network, and the accuracy of that model can lead to better solutions either in time complexity, nearness to optimality, or both.

The chapter concludes by showing the graph coloring problem traditionally used in solving network scheduling (undirected vertex distance–2 coloring) introduces non–existent dependencies between transceivers. An alternative, directed coloring, is defined, and evidence is provided to support its use in solving network scheduling problems.

Broadcast scheduling using the new graph model is explored in chapter 4. The chapter begins by showing the complexity of solving both traditional coloring and directed coloring problems. When restricted to 1–dimension point graphs can be optimally solved in polynomial time. For dimensions greater than one, coloring of point graphs is NP–complete. Two types of algorithms are developed to determine network schedules. Geometric algorithms are developed to include the position information available with point graphs. The geometric algorithms represent an initial attempt to understand the relationship between the spatial distribution of network points and the operation of greedy coloring algorithms. The time complexity of the geometric algorithms was better than the other algorithms tested ($O(n \log n)$ and $O(n^2)$), but they used substantially more colors on average than the other algorithms. The second type of algorithm developed uses the interconnection topology of the network rather than the locations of the transceivers within the network. The algorithms using these heuristics as an ordering mechanism consistently provided better results than most other algorithms. The time complexity for these topological algorithms was $O(n^2)$. The performance of the developed algorithms is compared to the performance of more traditional coloring algorithms which have been modified to solve directed coloring problems. The traditional algorithms were all $O(n^3)$ in time complexity. The best of the traditional algorithms provided results comparable to the best of the developed algorithms.

The application of point graphs in link scheduling is explored in chapter 5. The complexity of prn–coloring is determined for point graphs, and approximate prn–coloring algorithms are developed to solve the link scheduling problem. As with broadcast scheduling, two types of algorithms were developed. A geometric algorithm was developed which ordered the links to be colored by their locations. The other algorithms developed in this work were topological algorithms which depended on the interconnection links between the vertices. All of the developed algorithms have a time complexity of $O(n^4)$. The performance of these algorithms is compared to the performance of algorithms proposed in recent research. The algorithms from recent research had a time complexity of $O(n^3 \log n)$. The topological algorithms outperformed all other algorithms in the test cases run.

Chapter 6 explores the issues associated with distributed scheduling in radio networks. An algorithm for broadcast scheduling in a mobile transceiver environment is developed which incorporates the results obtained for centralized scheduling algorithms. This algorithm is evaluated informally for correctness. Efficiency and time complexity of this algorithm were not addressed in this work.

In conclusion, this work explored the problem of radio network scheduling using graph models. It was previously demonstrated that this problem mapped well to graph coloring problems which were shown to be NP–complete. The models previously used were shown to be inaccurate in their representation of the important characteristics of the problem domain, and a new model was defined which accurately represented the characteristics important to the problem solution. This new problem was also shown to be NP–complete in specific cases, and approximation algorithms were developed to solve scheduling using the new model. The main contributions of this dissertation include:

- The *Point Graph*, a new, more accurate graph model for radio networks is defined, after showing that previous models failed to represent the important features of radio networks. All network graph models are constructed from the same initial information. The identification of the model as a particular graph type determines many of the properties of solving coloring for the graph model. Previous models were either two restrictive and incapable of representing the interconnections of the network, or were not restrictive enough and modeled networks with no physical realization. Using an inaccurate model is the equivalent of solving the wrong problem. This new model is a new type of graph which is a superset of unit disk graphs and space graphs, both of which have been studied in the literature. This new model is based on the geometry of the radio network. An accurate model can lead to better solutions in either the time complexity, the closeness to optimality, or both.

- A new graph coloring problem is defined for application in the area of network scheduling. The inefficiency of traditional graph coloring for undirected graphs is explained, and experimentally demonstrated. The new graph problem of directed coloring is shown to use significantly fewer (20% less) colors in solving the broadcast scheduling problem. It is also shown that directed coloring will never require more colors than undirected coloring to color a graph.

- Complexity results for the new model are equivalent to the problem of network scheduling. Previous proofs used inaccurate network models in obtaining the complexity results. Therefore, though strongly indicating the complexity of network broadcast scheduling, the previous work was not directly applicable to radio networks and did not provide any proof of the actual complexity. The NP–completeness of distance–1 and distance–2 coloring N–dimensional point graphs was proven for N $\geq$ 2. Optimal polynomial time algorithms were given for coloring linear point graphs.

- The complexity of prn–coloring of networks is shown to be NP–complete for point graphs of dimension greater than one. As with broadcast scheduling, previous results were applicable to the models used to represent the networks. The time complexity of 1–dimensional point graphs is an open problem.

- Algorithms were developed which addressed the geometry of the network in the heuristics used in graph coloring. Since previous models were not specifically based on network geometry, this approach to graph coloring had never been explored. The specific algorithms developed in this work were based in part on the observation that linear point graphs can be optimally colored in $O(n \log n)$ time. Planar point graphs were projected to a line and colored as if they were linear point graphs. The loss of information in projecting the points was expected to produce poor results, but the algorithm for coloring had a $O(n \log n)$ time complexity. Two other algorithms used the projection of points to a line as a means of ordering the vertices to determine a greedy coloring of the graph. More complex heuristics addressing the relative density of points in the network may prove more useful in ordering the vertices. The time complexity of these algorithms is better than other algorithms, but they produce less efficient schedules.

- Development of realistic experimental models and the comparison of previously developed coloring algorithms and heuristics to those developed in this work.

7.2 Directions for Future Research

Several issue remain in the application area of network scheduling.

Point Graphs: These were defined in this work, but only explored in the realm of graph coloring. Graph theoreticians now have a new type of graph to consider. Point graphs are a superset of both unit disk graphs and space graphs and there are bound to be some similarities (particularly for symmetric point graphs). However, the complete analysis of this class of graphs remains to be done. Some specific graph problems which need to be explored include: forbidden subgraph characterizations; the recognition problem for point graphs; maximum independent set; and network flow problems. Some forbidden graphs were defined in chapter 2, but a complete characterization was not a goal of this work. Another issue in graph theory is the graph recognition problem, which is one of the areas where forbidden subgraphs are applied. The maximum independent set is a problem of interest to network designers as the maximum number of transceivers which can transmit at the same time. Network flow problems would also be of interest to network designers as a method of determining some of the measures of network capacity and throughput. Two graph coloring problems remain open. Chapter 4 contained proofs of the NP–completeness of distance–2 vertex coloring for non-uniform range point graphs, but no corresponding proof was provided for uniform range (symmetric) point graphs. In particular, symmetric planar point graph k–colorability for $3 < k < 7$ is an open problem. Also prn–coloring of linear point graphs is an open problem, though it is conjectured that it is NP–complete.

Graph Coloring: This research explored geometric algorithms and classical coloring heuristics in developing algorithms to solve the network scheduling problem. Further research should be done to determine what combination of topological and geometric characteristics is most important in developing heuristics for approximate graph coloring algorithms. The use of the locations of points gives more information than an adjacency matrix. Can this information be used to improve the time complexity or quality of solution of coloring algorithms, or is some more traditional idea consistently better at indicating the relative order of a node compared to its neighbors. For example, the degree of a given node is usually a good indicator of the impact a given node will have on the coloring of the network. It is also clear that degree information alone is insufficient in determining the best order to color nodes to achieve near optimal results. There is also no theoretical work indicating whether it is possible to determine a perfect ordering for the nodes to be colored. It was shown that a linear point graph can be optimally colored by proceeding through the vertices from, say, the minimum location to the maximum location. This raises the question, is it possible to order vertices in multi–dimension point graphs such that a greedy coloring algorithm will produce an optimal answer? Or can it be shown that there are graphs for which no such ordering can exist?

Distributed Scheduling: The work on distributed scheduling in this dissertation barely scratched the surface of the work which needs to be done. In particular, issues such as schedule coordination, node failures, and the addition of new nodes must be further explored, and a rigorous theoretical model must be developed. This model will permit more intelligent design decisions to be made when constructing a network.

Networks of Mobile Transceivers: A single distributed algorithm was presented in this work to address this problem. The changing topology of networks composed of mobile transceivers makes it necessary to reschedule the network as the topology changes. As communication links are formed and lost, the schedule will contain slots for links which no longer exist and will not have slots for recently formed links. How does this impact the scheduling process? In broadcast scheduling it will sometimes be possible to predict the creation of a new communication link due the distance–2 relationship which will likely exist between the nodes prior to the creation of the new communications link. Research should address the relationship between the frequency of link formation/dissolution and network rescheduling. Also the impact of mobility on optimality concerns of scheduling should be investigated.

Bibliography

[1] Arikan, Erdal. "Some Complexity Results about Packet Radio Networks," *IEEE Transactions on Information Theory*, *IT-30*(4):681–685 (July 1984).

[2] Baker, Brenda S. "Approximation Algorithms for NP–Complete Problems on Planar Graphs," *Journal of the Association for Computing Machinery*, *41*(1):153–180 (January 1994).

[3] Baker, Dennis J. and Anthony Ephremides. "The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm," *IEEE Transactions on Communications*, *COM-29*(11):1694–1701 (November 1981).

[4] Bar-Yehuda, Reuven, et al. "On the Time–Complexity of Broadcast in Multi–hop Radio Networks: An Exponential Gap between Determinism and Randomization," *Journal of Computer and System Sciences*, *45*:104–126 (1992).

[5] Bar-Yehuda, Reuven, et al. "Multiple Communication in Multi-hop Radio Networks." In *Eighth Annual ACM Symposium on Princ. Distrib. Comput.*, pages 329–338, 1989.

[6] Bećа, Hajrudin O., et al. "A Design Concept for Reconfigurable Mobile Radio Networks with Slow Frequency–Hopping Signaling," *IEEE Journal on Selected Areas in Communications*, *8*(4):603–612 (May 1990).

[7] Bertsekas, Dimitri and Robert Gallager. *Data Networks* (1 Edition). Englewood Cliffs, New Jersey: Prentice–Hall, Inc., 1987.

[8] Biedl, Therese and Goos Kant. "A Better Heuristic for Orthogonal Graph Drawing." An extended abstract submitted., 1994.

[9] Blum, Avrim. "New Approximation Algorithms for Graph Coloring," *Journal of the Association for Computing Machinery*, *41*(3):470–516 (May 1994).

[10] Bondy, J. A. "Bounds for the Chromatic Number of a Graph," *Journal of Combinatorial Theory*, *7*:96–98 (1969).

[11] Burr, Stefan A. "An NP–Complete Problem in Euclidean Ramsey Theory." In *Congressus Numerantium*, Volume 35, pages 131–138, December 1982.

[12] Chvátal, V., et al. "Four Classes of Perfectly Orderable Graphs," *Journal of Graph Theory*, *11*(4):481–495 (1987).

[13] Cidon, Israel and Moshe Sidi. "Distributed Assignment Algorithms for Multihop Packet Radio Networks," *IEEE Transactions on Computers*, *38*(10):1353–1361 (October 1989).

[14] Clark, A. P. *Principles of Digital Data Transmission* (Second Edition). New York: John Wiley & Sons, 1983.

[15] Clark, Brent N., et al. "Unit Disk Graphs," *Discrete Mathematics*, *86*:165–177 (1990).

[16] Dailey, David P. "Uniqueness of Colorability and Colorability of Planar 4–Regular Graphs are NP–Complete," *Discrete Mathematics*, *30*:289–293 (1980).

[17] Dirac, G. A. "The Structure of k–chromatic Graphs," *Fundamenta Mathematicae*, *40*:42–55 (1953).

[18] Edwards, Keith. "The complexity of some graph colouring problems," *Discrete Applied Mathematics*, *36*:131–140 (1992).

[19] Ehrlich, G., et al. "Intersection Graphs of Curves in the Plane," *Journal of Combinatorial Theory*, *21*:8–20 (1976).

[20] Ephremides, Anthony and Thuan V. Truong. "Scheduling Broadcasts in Multihop Radio Networks," *IEEE Transactions on Communications*, *38*(4):456–461 (April 1990).

[21] Even, S., et al. "On the NP–Completeness of Certain Network Testing Problems," *Networks*, *14*:1–24 (1984).

[22] Garey, Michael R., et al. "The Complexity of Computing Steiner Minimal Trees," *SIAM Journal of Applied Mathematics*, *32*(4):835–859 (June 1977).

[23] Garey, Michael R. and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[24] Garey, Michael R., et al. "Some Simimpified *NP*–Complete Graph Problems," *Theoretical Computer Science*, *1*:237–267 (1976).

[25] Gilmore, P. C. and A. J. Hoffman. "A Characterization o Comparability Graphs and of Interval Graphs," *Math and Computer Science*, pages 539–548 (1963).

[26] Gionfriddo, Mario. "A Short Survey on Some Generalized Colourings of Graphs," *ARS Combinatoria*, *24B*:155–163 (1987).

[27] Gupta, Ram Prakash. "Bounds on the Chromatic and Achromatic Numbers of Complementary Graphs," *Recent Progress in Combinatorics*, pages 229–235 (1969).

[28] Gupta, Udaiprakash I., et al. "An Optimal Solution for the Channel–Assignment Problem," *IEEE Transactions on Computers*, *C-28*(11):807–810 (November 1979).

[29] Gupta, Udaiprakash I., et al. "Efficient Algorithms for Interval Graphs and Circular–Arc Graphs," *Networks*, *12*:459–467 (1982).

[30] Hale, William K. "Frequency Assignment: Theory and Applications," *Proceedings of the IEEE*, *68*(12):1497–1514 (December 1980).

[31] Hansen, Pierre and Michel Delattre. "Complete–Link Cluster Analysis by Graph Coloring," *Journal of the American Statistical Association*, *73*(362):397–403 (June 1978).

[32] Holyer, Ian. "The NP–Completeness of Edge–Coloring," *SIAM Journal on Computing*, *10*(4):718–720 (November 1981).

[33] Hu, Limin. *Distributed Algorithms for Packet Radio Networks.* Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, California, September 1990.

[34] Hu, Limin. "Distributed Code Assignments for CDMA Packet Radio Networks." In *INFOCOM*, pages 1500–1509, 1991. (pp. 12D.3.1–12D.3.10).

[35] Hu, Limin. "Topology Control for Multihop Packet Radio Networks," *IEEE Transactions on Communications*, *41*(10):1474–1481 (October 1993).

[36] Huson, Mark L. and Arunabha Sen. "Broadcast Scheduling Algorithms for Radio Networks." In *MILCOM '95*, 1995. Accepted for publication, MILCOM '95, to be held November 1995.

[37] Huson, Mark L. and Arunabha Sen. "Graph Paradigms for Multi–hop Radio Networks." In *GLOBECOM '95*, 1995. Submitted for publication Feb '95.

[38] Huson, Mark L. and Arunabha Sen. "Scheduling Algorithms for Packet Radio Networks." In *ICCC '95*, 1995. Accepted for publication, ICCC'95, to be held August 1995.

[39] Jaromczyk, Jerzy W. and Godfried T. Toussaint. "Relative Neighborhood Graphs and Their Relatives," *Proceedings of the IEEE*, *80*(9):1502–1517 (September 1992).

[40] Lee, D. T. "Maximum Clique Problem of Rectangle Graphs," *Advances in Computing Research*, *1*:91–107 (1983).

[41] Lekkerkerker, C. G. and J. Ch. Boland. "Representation of a finite graph by a set of intervals on the real line," *Fundamenta Mathematicae*, pages 45–64 (1962).

[42] Lynch, Clifford A. and Edwin B. Brownrigg. *Packet Radio Networks: Architectures, Protocols, Technologies and Applications* (1 Edition). Elmsford, NewYork: Pergamon Press, 1987.

[43] Maehara, Hiroshi. "Space Graphs and Sphericity," *Discrete Applied Mathematics*, *7*:55–64 (1984).

[44] Olariu, Stephan. "An optimal greedy heuristic to color interval graphs," *Information Processing Letters*, *37*:21–25 (1991).

[45] Orponen, Pekka, et al. "Instance Complexity," *Journal of the Association for Computing Machinery*, *41*(1):96–121 (January 1994).

[46] Ramanathan, S. and Errol L. Lloyd. *An algorithmic study of certain broadcast network problems*. Dept. of Computer and Information Sciences TR 92–19, Newark, Delaware 19716: University of Delaware, 1992.

[47] Ramanathan, S. and Errol L. Lloyd. "Scheduling Algorithms for Multi–hop Radio Networks." In *SIGCOM*, pages 211–222, 1992.

[48] Ramanathan, Subramanian. *Scheduling algorithms for multi-hop radio networks*. Department of Computer and Information Sciences, University of Delaware, Newark, Delaware, December 1992.

[49] Ramanathan, Subramanian and Errol L. Lloyd. *Complexity of certain graph coloring problems with applications to radio networks*. Dept. of Computer and Information Sciences TR 92–18, Newark, Delaware 19716: University of Delaware, 1992.

[50] Ramanathan, Subramanian and Errol L. Lloyd. "Efficient Distributed Algorithms for Channel Assignment in Multihop Radio Networks." Submitted to Journal of High Speed Networks, 1993.

[51] Ramaswami, Rajiv and Keshab K. Parhi. "Distributed Scheduling of Broadcasts in a Radio Network." In *INFOCOM*, pages 497–504, 1989.

[52] Roberts, Fred S. "On the Boxicity and Cubicity of a Graph," *Recent Progress in Combinatorics*, pages 301–310 (1969).

[53] Rustad, John Erik, et al. "New Radio Networks for Tactial Communication," *IEEE Journal on Selected Areas in Communications*, *8*(5):713–727 (June 1990).

[54] Segall, Adrian. "Distributed Network Protocols," *IEEE Transactions on Information Theory*, *IT-29*(1):23–35 (January 1983).

[55] Sousa, Elvino S. and John A. Silvester. "Spreading Code Protocols for Distributed Spread–Spectrum Packet Radio Networks," *IEEE Transactions on Communications*, *36*(3):272–281 (March 1988).

[56] Tobagi, Fouad A. and Leonard Kleinrock. "Packet Switching in Radio Channels: Part II–The Hidden Terminal Problem in Carrier Sense Multiple–Access and the Busy–Tone Solution," *IEEE Transactions on Communications*, *COM-23*(12):1417–1433 (December 1975).

[57] Valiant, Leslie G. "Universality Considerations in VLSI Circuits," *IEEE Transactions on Computers*, *c–30*(2):135–140 (February 1981).